(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－312220
（P2002－312220A）
(43)公開日　平成14年10月25日(2002.10.25)

(51)Int.Cl.[7]		識別記号	FI		テーマコード（参考）
Ｇ０６Ｆ	12/00	５３７	Ｇ０６Ｆ	12/00	５３７Ａ　５Ｂ０７５
	17/30	１２０		17/30	１２０Ｂ　５Ｂ０８２
		１８０			１８０Ｄ

審査請求　未請求　請求項の数20　ＯＬ　外国語出願　（全 45 頁）

(21)出願番号　特願2002－45438(P2002－45438)

(22)出願日　平成14年１月18日(2002.1.18)

(31)優先権主張番号　６０／２６３１１１
(32)優先日　平成13年１月18日(2001.1.18)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０９／８１６６４０
(32)優先日　平成13年３月22日(2001.3.22)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　藤原　真二
アメリカ合衆国、カリフォルニア州 95014、クペルティノ　＃4300、プルネリッジ・アベニュー　19500
(72)発明者　ジェームス・ロコウィッツ
アメリカ合衆国、イリノイ州　60074、パラティン　＃212、ウエスト・ヒックス 1971
(74)代理人　100080001
弁理士　筒井　大和

最終頁に続く

(54)【発明の名称】　ユーザ定義機能を使用したセルレベルのデータアクセス制御

(57)【要約】
【課題】　アクセス手段に基づいて、柔軟なセルレベルデータのアクセス制御技術の必要性が存在する。システムコストを低減できるアクセス手段の実行が必要とされる。
【解決手段】　セルレベルでのアクセス制御がマスク機能１２２２の使用によって提供される。アクセス手段に従って制御されるアクセスが望まれるセル用のマスク機能１２２２を含むように、本来の問い合わせ１１０２が変更される。更に、フィルタ機能１２０４は、アクセス手段に従って行を削除するために含まれる。

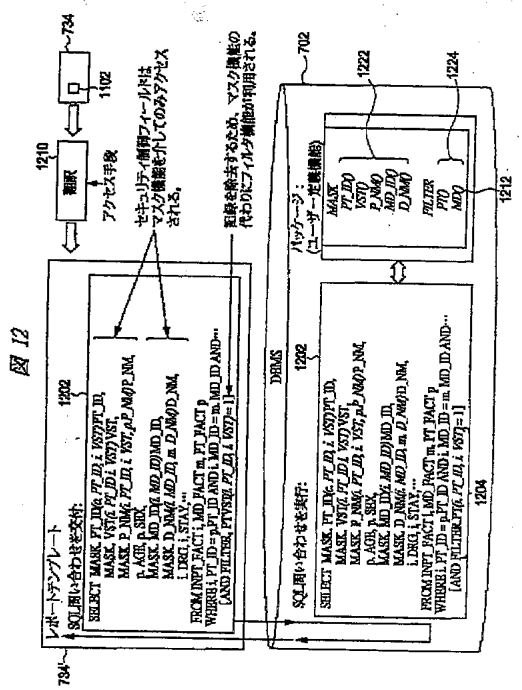


図 12

【特許請求の範囲】

【請求項１】　アクセス手段に従って情報記憶装置内の情報にアクセスする方法であって、

第１タイプ情報用のリクエストからなるアクセスリクエストを受け取り、それと前記情報記憶装置内に含まれる第１情報を前記第１タイプ情報用のリクエストが関連づけ、

前記第１タイプ情報用のリクエストを、前記アクセス手段に基づく第１タイプ情報用の変更リクエストと置き替え、

前記情報記憶装置にアクセスして、前記アクセスリクエストに応答して結果を作成し、前記変更リクエストが、前記アクセス手段に基づいて、マスクされた値、又は、前記第１情報を作成することを特徴とするアクセス方法。

【請求項２】　請求項１記載のアクセス方法において、前記変更リクエストがマスク機能を備えていることを特徴とするアクセス方法。

【請求項３】　請求項２記載のアクセス方法において、前記情報記憶装置にアクセスする工程が、前記マスク機能を実行して前記マスクされた値、又は、前記第１情報の何れかを作成することを含むことを特徴とするアクセス方法。

【請求項４】　請求項１記載のアクセス方法において、前記アクセスリクエストを変更して、前記アクセス手段に従って前記結果の一部分を削除するために効果的なフィルタ機能を備えることを更に有することを特徴とするアクセス方法。

【請求項５】　請求項１記載のアクセス方法において、前記情報記憶装置が関係型データベースであり、

前記第１タイプ情報用のリクエストがＳＥＬＥＣＴ文からなり、

前記ＳＥＬＥＣＴ文が一つ以上の列基準からなり、

前記変更リクエストが、前記一つ以上の列基準のうち少なくとも一つをマスク機能と置き替えることからなることを特徴とするアクセス方法。

【請求項６】　請求項１記載のアクセス方法において、前記情報記憶装置が関係型データベースであり、

前記アクセスリクエストがＷＨＥＲＥクローズを含み、

前記結果が情報の一つ以上の行からなり、

前記アクセス方法が、前記アクセス手段に基づいて、前記ＷＨＥＲＥクローズの中のフィルタ機能を組み入れて前記結果内に含まれる幾つかの行を除去することを更に有することを特徴とするアクセス方法。

【請求項７】　関係型のデータベース内で、アクセス手段に従って情報にアクセスする方法であって、

一つ以上の列基準からなるＳＥＬＥＣＴ文を有する少なくとも一つの問い合わせを与え、

前記一つ以上の列基準のうち少なくとも一つをマスク機能と置き替えて変更問い合わせを作成し、

情報の一つ以上の行からなる前記変更問い合わせに応答して問い合わせ結果を作成し、

前記問い合わせ結果が、前記アクセス手段に基づいて、前記一つ以上の列基準のうち前記少なくとも一つに対し、前記関係型データベースからのマスク値又は情報の何れかを含むことを特徴とするアクセス方法。

【請求項８】　請求項７記載のアクセス方法において、前記少なくとも一つの問い合わせが更にＷＨＥＲＥクローズからなり、

前記アクセス方法が、前記ＷＨＥＲＥクローズを変更してフィルタ機能を含んだ変更ＷＨＥＲＥクローズを作成することを更に有し、

前記フィルタ機能が二つの論理値のうち一方を作成し、

前記変更ＷＨＥＲＥクローズが、前記フィルタ機能によって作成される値に基づく前記問い合わせ結果から行を削除するために効果的であることを特徴とするアクセス方法。

【請求項９】　請求項７記載のアクセス方法において、前記関係型データベースがデータベースサーバー内に設けられ、

前記少なくとも一つの問い合わせ与える工程が、クライアントシステムで前記少なくとも一つの問い合わせを受け取ることを含み、

前記問い合わせ結果を作成する工程が、前記変更問い合わせを前記データベースサーバーに送ることを含むことを特徴とするアクセス方法。

【請求項１０】　請求項９記載のアクセス方法において、

前記少なくとも一つを置き替える工程が前記クライアントシステムで実行されることを特徴とするアクセス方法。

【請求項１１】　請求項９記載のアクセス方法において、

前記少なくとも一つを置き換える工程が前記データベースサーバーで実行されることを特徴とするアクセス方法。

【請求項１２】　アクセス手段に従って情報記憶装置にアクセスするために、内部に含まれたコンピュータ読み出し自在なプログラムコードを有するコンピュータメモリを備えたコンピュータによる情報検索システムにおいて、

第１タイプ情報用のアクセスリクエストを受け取るように構成され、前記第１タイプ情報用のアクセスリクエストにより第１情報が関係づけられる第１コードと、

前記第１タイプ情報用のリクエストを、前記アクセス手段に基づいた第１タイプ情報用の変更リクエストと置き替えるように構成された第２コードと、

前記情報記憶装置にアクセスして、前記アクセスリクエストに応答して結果を作成するように構成された第３コードとを備え、

前記変更リクエストが、前記アクセス手段に基づいて、マクスされた値、又は、前記第１情報の何れか一方を作成することを特徴とするコンピュータによる情報検索システム。

【請求項１３】　請求項１２記載のコンピュータによる情報検索システムにおいて、
前記アクセスリクエストを変更して、前記アクセス手段に従って前記結果の一部分を削除するのに効果的なフィルタ機能を含むように構成された第４コードを更に備えていることを特徴とするコンピュータによる情報検索システム。

【請求項１４】　請求項１２記載のコンピュータによる情報検索システムにおいて、
関係型データベースを更に備え、
前記第１タイプ情報用のリクエストが、一つ以上の列基準からなるＳＥＬＥＣＴ文を有し、
前記変更リクエストが、前記一つ以上の列基準のうち少なくとも一つをマスク機能と置き替えることを含むことを特徴とするコンピュータによる情報検索システム。

【請求項１５】　請求項１２記載のコンピュータによる情報検索システムにおいて、
関係型データベースを更に備え、
前記アクセスリクエストがＷＨＥＲＥクローズを含み、
前記結果が、情報の一つ以上の行からなり、
前記第２コードが、前記アクセス手段に基づいて、前記ＷＨＥＲＥクローズ内にフィルタ機能を組み込んで前記結果の中に含まれる幾つかの行を削除するように更に構成されていることを特徴とするコンピュータによる情報検索システム。

【請求項１６】　請求項１２記載のコンピュータによる情報検索システムにおいて、
クライアントコンピュータシステムとサーバーコンピュータシステムとを更に備え、
前記クライアントコンピュータシステムが、前記第１及び第２コードを含む前記コンピュータメモリの一部分からなり、
前記サーバーコンピュータシステムが、前記第３コードを含む前記コンピュータメモリの別の一部分からなることを特徴とするコンピュータによる情報検索システム。

【請求項１７】　請求項１２記載のコンピュータによる情報検索システムにおいて、
前記データベースサーバーが関係型データベースサーバーであり、
前記第１タイプ情報用のリクエストが、一つ以上の列基準からなるＳＥＬＥＣＴ文を有し、
前記変更リクエストが、前記一つ以上の列基準のうち少なくとも一つを、マスク機能と置き替えることを備えていることを特徴とするコンピュータによる情報検索システム。

【請求項１８】　請求項１７記載のコンピュータによる情報検索システムにおいて、
前記第３コードがマスク機能を有することを特徴とするコンピュータによる情報検索システム。

【請求項１９】　請求項１６記載のコンピュータによる情報検索システムにおいて、
前記データベースサーバーが関係型データベースサーバーであり、
前記アクセスリクエストがＷＨＥＲＥクローズを含み
前記結果が情報の一つ以上の行からなり、
前記第２コードが、前記アクセス手段に基づいて、前記ＷＨＥＲＥクローズの中にフィルタ機能を組み込んで前記結果に含まれる幾つかの行を削除するように更に構成されていることを特徴とするコンピュータによる情報検索システム。

【請求項２０】　請求項１９記載のコンピュータによる情報検索システムにおいて、
前記第３コードがマスク機能を有することを特徴とするコンピュータによる情報検索システム。

【発明の詳細な説明】

【０００１】
【発明の属する技術分野】本発明は、データベースのアクセスに関し、特に、データベース内のフィールドへの制御されたアクセスに関する。

【０００２】
【従来の技術】今日の情報技術によって、様々な種類のデータソースへのシームレスアクセスが体験可能となっている。そのような技術のおかげで、益々増える情報を手に入れやすくなっている。しかしながら、データソースには、認可されないアクセスから保護されるべき医療記録、財務記録、及び他の類似する個人情報のような重要情報がしばしば含まれ、そのような情報にアクセスしたい人々のアクセス権（access privilege）が必要とされている。データベースシステムが発展すると、ビュー定義及び認可モデルを使った一組のデータアクセス制御機能が提供される。

【０００３】ビューとは情報対象であり、それによって通常の表内であるが、異なる方法でデータを見ることが可能となる。それは、データベースを構成する一定の表の一部分からなる機能上論理的に定義された表である。ビューは、データの合成を必要とせずに、基本的な表のデータを見る方法を提供する。

【０００４】伝統的なビューは、行レベル及び／又は列レベル方式でデータベース内のデータへのアクセスを制御できる。図１は病院データＩＮＰＴ＿ＢＡＳＥ１００の一例を示し、それは入院患者情報と、ＭＤ＿ＩＤによって分類され集められた入院患者情報とを含んでいる。各々の医者は、自己の患者を往診に行くためにだけ、そのデータを見ることが許可されていると仮定する。図２は、各々の医者のためのＩＮＰＴ＿ＢＡＳＥ１００における所望のビューを示している。ＰＴ＿ＩＤ、ＶＳＴ、

Ｐ＿ＮＭ、及び、ＭＤ＿ＩＤフィールドは、各患者のプライバシーを保護するために選択的に目に見えなくされているので、医者は自分の患者用のデータだけ見ることができる。こうして、ＩＤが２２２２である医者にとって、自己に利用可能であるべきビューはビュー２０２となる。ＩＤが３３３３である医者にとって、ビューはビュー２０４となる。

【０００５】入院患者用のビューは、従来のビュー定義（又はビュー作成）によって定義され得る。例えば、図３は、図２で示されるビュー２０２，２０４，及び２０６を作成するビュー定義を示している。（なお、オラクル（Ｏｒａｃｌｅ）データベースシステムの場合には、ｕｓｅｒ＿ｉｄは、最新のｕｓｅｒ＿ｉｄに戻す表現、即ち、ＳＹＳ＿ＣＯＮＴＥＸＴ（「ｕｓｅｒｅｎｖ」、「ｓｅｓｓｉｏｎ＿ｕｓｅｒ」）と置き替え可能である。）しかしながら、図４のＳＱＬ文を実行して、ＭＤ＿ＩＤにより分類され集められる入院患者情報を得る場合には、各々の医者は図５で示されるような異なった結果を得るだろう。図２で示されるような所望の集約結果を得るために、図６で示されるビューを定義できる。しかしながら、特別な多重次元分析を許可するために、集約ビューのあらゆる可能な組合せを定義しなければならない。このような物理的な力によるアプローチは、大部分、ビューメンテナンス費用を著しく増大させる。例えば、医者が特別なＤＲＧ（診断に役立つ関連グループ）の統計、即ちＤＲＧ　ＢＥＴＷＥＥＮ１２０、１２９を見たい場合には、別途、ＭＤ＿ＩＤによって分類されるデータの部分集合を集約するビューを定義しなければならない。各々の医者が、異なる部分集合のデータを見たい場合には、このようなビューを予め準備することは殆ど不可能である。

【０００６】

【発明が解決しようとする課題】最近のシステムでは、アプリケーション理論の一部としてアクセス制御手段を実行することによってこの問題が解決されている。しかしながら、一般的なシステムでは多数のアプリケーションが存在する。その結果として、異なるアプリケーションの各々の中で一つのアクセス手段が実行されなければならず、タスクがアクセス手段のメンテナンス費用を著しく増大させる。レガシーソフトウェアを利用し続ける場合には、その努力成果は完全にくじかれるだろう。

【０００７】フロー、推論、及び、アクセス制御を含めた様々なセキュリティ手段を通してデータベースを保護することができる。情報システム内のアクセス制御は、保護手段によって決定されるモデル及びルールに従って、システム対象へのあらゆるダイレクトアクセスが排他的に生じることを請け合うことの原因となる。アクセス制御は、データベースシステム用のコンテンツ依存アクセス制御モデルまで高められている。コンテンツ依存アクセス制御モデルに基づく従来のビュー定義では、アクセスルールは、（ｓ，ｏ，ｔ，ｐ）の組によって表示され、それは、プリディケイトｐが当てはまるオブジェクトｏの発生へのアクセスｔをサブジェクトｓが有することを明記している。そのモデルの強化は、６組の（ａ，ｓ，ｏ，ｔ，ｐ，ｆ）からなり、ａは、ｓに権利（ｏ，ｔ，ｐ）を与える権限者のサブジェクトであり、そして一方、ｆは、ｓが（ｏ，ｔ，ｐ）を他のオブジェクトに更に移す可能性を記述するフラッグのコピーである。

【０００８】多くのセキュリティモデルは従来技術文献において提案されている。アクセスマトリックス（Access Matrix）モデル、テイクーグラアント（Take-Grant）モデル、アクション－エンディティ（Action-Entity）モデル、及び、ウーデン・エトオール（Wood et at.）モデルは、任意のセキュリティモデルである。ユーザの問い合わせが使用許可と照会される。それが許可されると、問い合わせが特別なアクセスモード内の対象にアクセスする。さもなければ、そのアクセスは否認される。

【０００９】T. F. Lunt、D. Denning、R. R. Schell、M. Heckman、及び、W. R. Shckleyにより、ソフトウェアに関するＩＥＥＥ会報、第１６巻No. 6（１９９０年６月）５９３頁～６０７頁、「シービュー・セキュリティモデル」と表題を付けられた論文には、シービュー・モデルとして公知なセキュリティモデル、即ち、マンデトリー・アクセス・コントロール（Mandatory Access Control，ＭＡＣ）モデル及びトラスト・コンピューティング・ベース（Trusted Computing Base，ＴＣＢ）モデルが提案され、二層の使用によって関係型データベースシステムのセキュリティを保護している。シービューは、物理的な単一レベル関係からビジュアル的な多重レベル関係の例を発生させることによって、多重レベルデータアクセスを制御する。

【００１０】他のモデルは、多重レベルのセキュリティモデルとして提案されたジェイジョディアーサンドヒュ・モデル（Jajodia-Sandhu's model）とスミスーウィンスレット・モデル（Smith-Winslett's model）とを含んでいる。これらのモデルのセキュリティ手段はビジュアル的な多重レベル関係の例を生成する。これらのモデルは、データベースシステムとアプリケーションとの間に配置されてデータベースのセキュリティを実行する代替フィルタを使用している。

【００１１】従来のビューを処理することは、次の一般的な工程を含んでいる。：

（１）オーセンティケーション（情報にアクセスする資格の有無を検証する。）

（２）ビュー定義を適用、即ちビュー定義に従って問い合わせを改訂する。

（３）その問い合わせを最適化する。

（４）その問い合わせを実行する。

（５）結果をリターンする。

【００１２】従来のビューでは、アクセス制御のルールは実行前に問い合わせに適用される。その問い合わせは、プロジェクション列の要素でない列にアクセスできない。更に、列の値をプロジェクションの対象として隠す機能をユーザが定義すると、問い合わせは本来の（オリジナルな）値もアクセスすることができない。

【００１３】情報システムセキュリティ会報２．１（１９９９年2月）、３４頁～６４頁、「企業イントラネット内での役割に基くアクセス制御モデルと基準方法」と表題を付けられた論文では、David F. Ferraiolo、John F. Barkely、及び、RichardD. Kuhnが、ユーザ役割のコンセプトに基づいて、アクセス権を付与する役割に基くアクセス制御を開示している。

【００１４】オラクル８ｉシステムは、バーチャルなプライベートデータベースを使った微粒子アクセスを有し、それは、オラクルマガジン（１９９９年7月）、「オラクル８ｉを備えたビジュアルなプライベートデータベースの作成」と表題を付けられ、Mary A. Davidsonによる公式報告書の中で議論されている。この機能によって、ユーザが表にアクセスする度に自動的に、データベース設計者が選択条件手段を付け足し可能となる。任意の値、例えば、コンテクスト値やセッション値に基づいてその条件手段を生成できる。しかしながら、条件は、その値を満たさない行を除去するので、１行内に列の小さな一組をマスクする（覆う）ことはできない。

【００１５】セキュリティモデルは、ＡＣＭ会報データベースシステム３．１（１９７８年３月）、９２頁～１０４頁、「小さなカウントを備えた問い合わせ用の統計データベースのセキュリティ」と表題を付けられ、F. Y. Chinによる論文で、統計上の推定を防止する統計データベースシステム用に提案されている。推定保護用の３つの技術、即ち、概念的な、制約に基づき、且つ、摂動に基づく技術がある。例えば、アディソン・ウェズリー出版社（１９９４年）、Castano、Silvana、 Fugini、 G. Mariagrazia、Martella、Giancarlo、Samarati、及びPierangelaによる「データベースセキュリティ」、とACM Computing Surveys、第２１巻No. ４（１９８９年１２月）、５１５頁～５５６頁、「統計データベース：比較検討」と表題を付けられ、Adam、R. Nabil、及びJohn C. Worthmannによる論文とを参照。これらの技術は、統計上の値を抑制し、又は、グループ範囲の組合せを制限している。しかしながら、その技術は、範囲の値自体を抑制する機能を提供していない。それ故に、それらは、図２に示されるような集約結果用のアクセス手段を定義できない。

【００１６】アクセス手段に基づいて、柔軟なセルレベルデータのアクセス制御技術の必要性が存在する。システムコストを低減できるアクセス手段の実行が必要とされる。

【００１７】

【課題を解決するための手段】本発明は、それぞれアクセス制御された列のためのマスク機能を使ってセルレベルのアクセス制御を提供する。各々のマスク機能は、アクセス許可を決定する一つ以上のキーパラメータに関連づけられている。マスク機能の中で具体化されるアクセス手段に依存して、マスク機能は、マスクされた列の値、又は、本来の列の値をリターンする。

【００１８】本発明の他の態様は、それぞれの行削除手段のためのフィルタ機能を使って、セルレベルのアクセス制御を提供する。各々のフィルタ機能は、一つ以上のキーパラメータに関連づけられている。フィルタ機能は２のカテゴリー（例えばバイナリー）値をリターンする。フィルタ機能のリターン値をチェックする条件は、問い合わせの条件クローズに加えられて、行削除手段に従って行を削除する。

【００１９】本発明の更に他の態様は、上述のセルレベルのアクセス制御メカニズムを提供するレポーティングシステムである。

【００２０】本発明が教えることは、添付図面と共に次の詳細な説明を検討することによって、容易に理解可能である。

【００２１】

【発明の実施の形態】図７に関して、本発明の一実施の形態を、ウェブに基づくレポーティングシステムのアーキテクチャ７００と共に説明する。アーキテクチャは三つのサーバー構成要素から成る。即ち、データベースサーバー７２２はデータベース管理システム（ＤＢＭＳ）７０２を含んでいる。ＤＢＭＳは任意の従来データベースにすることがある。一つの特別な実例の形態では、ＤＢＭＳは関係型データベースシステムである。レポートサーバー７０６は、従来の通信設備上でデータベースサーバーと繋がり、従来の通信設備の詳細特性は、本発明の特別な実施の形態に依存する。レポートサーバーは、複数のレポートテンプレート７３４を備えてレポート生成サーバーを提供する機能を促進する。ウェブサーバー７０４は、レポートサーバーに繋がり、ＤＢＭＳへのクライアント側のアクセスを提供する。ウェブサーバーは、従来の通信設備上でレポートサーバーに繋がり、従来の通信設備の詳細特性は本発明の特別な形態に依存する。

【００２２】図7は、サーバー構成要素における一般的なソフトウェア及びハードウェアの構成を示している。データベースサーバー７２２は、自身のコンピュータシステムを有し、大きい容量記憶のサブシステムを備えている。図示されたレポートサーバー７０６とウェブサーバー７０４とは、別のコンピュータシステム７３４に属し、データベースサーバーから分離している。実際には、ウェブサーバーとレポートサーバーとは、所望のスループットを達成するために、ウェブサーバー処理及び

レポートサーバー処理の多数のインスタンシエーション（instantiation）で構成することがある。なお、多くの代替構成が可能である。例えば、単一のコンピュータシステムを使って小さなスケール動作中に三つのサーバー構成要素を全てホストできる。大きな装置の中では、各サーバーが自分自身のコンピュータシステムを占有することがある。更に多くのスループットを提供するために、実際には、各サーバーは、非常に大きなシステム内で多数のサーバーシステムから成る。

【００２３】ＤＢＭＳ７０２へのユーザのアクセスは、クライアントのブラウザ７１２を介して行われ、更に第３コンピュータシステム７２６上で実行される。ブラウザは、ハイパーテキスト・トランスポート・プロトコル（ＨＴＴＰ）、又は、ＳＳＬプロトコル上のＨＴＴＰ（ＨＴＴＰＳ）を使って、ウェブサーバー７０４に繋がっている。

【００２４】ユーザはブラウザ７１２を介してウェブサーバー７０４と交互作用してレポートを得る。最初に、レポートテンプレート７３４を選択する。次に、テンプレート用の一組のパラメータを与える。ウェブサーバーは、ユーザ提供パラメータと一緒に、ユーザ選択レポートテンプレートに対応するユーザ識別子をレポートサーバー７０６に回す。レポートサーバーは、選択されたレポートテンプレートに関係される一つ以上の問い合わせを、データベースサーバー７２２に交付する。レポートサーバーとデータベースサーバーとの間での適切な交互作用の後に、その問い合わせ結果がレポートサーバーに戻される。レポートサーバーはその結果を受け取り、その当時にウェブサーバーを介してユーザに送付される体裁の良い形式にその結果をフォーマットする。

【００２５】図８は、説明目的のためにだけ示された簡単なデータアクセス手段の一例である。この例では、三つのアクセスレベル、即ち実行レベル、医療ドクターレベル、及び財務アナリストレベルが望まれていると仮定する。実行レベルのユーザは、全てのデータにアクセスすることを許可されている。一般的に、これは、行政官全職員とデータベース管理者全職員とのために体系化されている。

【００２６】医者は、医療ドクターレベルのユーザのアクセス権を与えられる。医者は、患者往診の処置に関する患者データと、医者が作成したデータとにアクセスできるべきである。しかしながら、医者は、患者のプライベート情報、例えばクレジットカード情報のうちのいくらかにアクセスすることを許可されていない。更に、医者は、別の医者のデータにアクセスすることを許可されていない。図８に記載された説明のためのアクセス手段によれば、医者は、他の医者によって処置された患者往診のために患者名を見ることができない。その医者が同一患者の他の往診を処置した場合であっても、患者名を見ることができない。例えば、医者２２２２が、図２における第１行の患者名を見ることができない。なぜならば、他の医者３３３３がＡＲＥＮの第１往診を処置しているからである。それ故に、図８で与えられた説明のためのアクセス手段によれば、たとえ医者２２２２がＡＲＥＮの第２往診を処置したとしても、その医者はＡＲＥＮの第１往診のためにその名前を見ることができない。理解されることは、そのような場合に、データへのアクセスを許可する他のアクセス手段が存在することである。更に理解されることは、本発明はそのようなアクセス手段を提供できることである。

【００２７】最後に、アクセス制御が、財務全職員に対して提供される。このクラスのユーザは、財務アナリストレベルのユーザアクセスを許される。財務アナリストは、患者の財務情報のうちいくらかを含んだ滞在、費用、及び支払のような財務情報にアクセスできる。しかしながら、財務アナリストは、医者によって必要とされる種類のデータへのアクセスをすることできない。

【００２８】図９に関して、関係型ＤＢＭＳ７０２（図７）用のデータスキーム９００の説明例が図示されている。ユーザ情報表９０２（ＵＳＥＲ＿ＩＮＦＯ）は、各ユーザ用のユーザ記録（例えば、ユーザ記録９１２）を含んでいる。それぞれの記録は、ユーザ関連情報９２６に加えて、ユーザｉｄフィールド９２２と役割フィールド９２４とを含んでいる。役割フィールドは、図８のアクセス手段毎に、各ユーザ用のアクセスレベルのアクセス権を識別する。

【００２９】入院患者情報表９０４（ＩＮＰＴ＿ＦＡＣＴ）は、患者に基づいて行われた各往診用の入院患者記録（例えば、入院患者記録９１４）を保持している。その結果として、患者は、各往診用に一つ、この表の中に多数の入力情報を持つ可能性が非常に高い。患者ｉｄフィールド９３１は患者を識別する。患者往診フィールド９３２（ＶＳＴ）は、一人の患者の各往診／入院発生を表している。別のフィールドは医療ドクターＩＤフィールド９３４であり、それは、処置した医者の識別子を含んでいる。

【００３０】患者情報表９０６（ＰＴ＿ＦＡＣＴ）は、各患者用の患者記録を含んでいる。各記録は、患者ｉｄフィールド９４２（ＰＴ＿ＩＤ）、患者名フィールド９４４（Ｐ＿ＮＭ）、及び患者性別フィールド９４６（ＳＥＸ）を含んでいる。類似の医者情報表９０８（ＭＤ＿ＦＡＣＴ）は各医者用の情報を含んでいる。これは、例えば、医療ドクターＩＤフィールド９５２（ＭＤ＿ＩＤ）、名前フィールド９５４（Ｄ＿ＮＭ）、及び医療ドクター部門フィールド９５６（ＤＥＰＴ）を含んでいる。

【００３１】図８及び図９に関して、それがデータスキーム９００に関係するようなアクセス手段の効果を説明する。例えば、役割ＩＩのユーザを検討する。役割ＩＩのユーザが医者であることが想起される。医者は、自分

で処置した患者だけのある情報を眺めることができるだけである。こうして、理解され得ることは、処置した医者は、患者年齢フィールド９３３、ＤＲＧフィールド９３５、滞在期間フィールド９３６、費用フィールド９３７、支払フィールド９３８、患者性別フィールド９４６、医療ドクター部門フィールド９５６を眺めることができる。しかしながら、患者往診が医者によって処置されない場合、又は、情報が医者自身のデータ（例えば、医者が自己の名前を見ることができる）である場合には、患者ｉｄフィールド９３１，９４２、患者往診フィールド９３２、患者名フィールド９４４、医療ドクターＩＤフィールド９３４，９５２、及び医療ドクター名フィールド９５４はその医者に利用可能であるべきない。従って、スキーム９００への問い合わせ結果が、問い合わせした医者によって処置された患者往診用の全データと、問い合わせした医者によって処置されなかった患者往診用の一部分マスクされたデータとを含むべきである。

【００３２】図８で示されるような役割ＩＩのユーザ用のアクセス手段は、患者往診によって（患者によってではない）ＰＴ＿ＩＤ、ＶＳＴ＿ＮＢＲ、及びＰ＿ＮＭのような患者プライベート情報へのアクセスを制限する。それ故に、患者プライベート情報をマスクするべきか否かを決定するためのキー組は列組｛ＰＴ＿ＩＤ、ＶＳＴ＿ＮＢＲ｝である。なぜならば、これらの列は、患者往診の対象にとって主要なキーになるからである。（アクセス手段が患者によってアクセスを制限している場合には、そのキー組は｛ＰＴ＿ＩＤ｝になる。）医療ドクター情報に関して、役割ＩＩのユーザは、自分自身のプライベート情報にアクセスするだけである。それ故に、それが自分自身のプライベート情報でない場合には、ＭＤ＿ＩＤとＤ＿ＮＭとは隠されるだろう。従って、医者のプライベート情報ＭＤ＿ＩＤ，ＤＮＭをマスクするべきか否かを決定するためのキー組は列組｛ＭＤ＿ＩＤ｝である。

【００３３】役割ＩＩのユーザがＳＥＬＥＣＴ*ＦＲＯＭ　ＰＴ＿ＦＡＣＴのような問い合わせを交付すると、全てのＰＴ＿ＩＤ及びＰ＿ＮＭ列を隠す（マスクする）べきである。なぜならば、役割ＩＩのユーザは病院内で患者リストを受け取るべきでないからである。役割ＩＩのユーザは自分自身の患者リストを作成だけできる。自分自身の患者リストを作成するために、次の問い合わせを交付するべきである：ＳＥＬＥＣＴ　ＤＩＳＴＩＮＣＴ　ａ．ＰＴ＿ＩＤ、ａ．Ｐ＿ＮＭ、ａ．ＳＥＸ　ＦＲＯＭ　ＰＴ＿ＦＡＣＴ　ａ．ＩＮＰＴ＿ＦＡＣＴ　ｂＷＨＥＲＥ　ａ．ＰＴ＿ＩＤ＝ｂ．ＰＴ＿ＩＤ、及び、ｂ．ＭＤ＿ＩＤ＝医者のｉｄ。この場合には、｛ＰＴ＿ＩＤ，ＶＳＴ＿ＮＢＲ｝の値を使うことによって列ＰＴ＿ＩＤ及びＰ＿ＮＭをマスクするべきか否かを決定することができる。なぜならば、その問い合わせがＰＴ＿ＦＡＣＴとＩＮＰＴ＿ＦＡＣＴとを結合するべきであるからである。要するに、マスクを決定する対象のキー列がその問い合わせ内の表によって覆われていない場合には、プライベートデータが見られることを許可しないだろう。

【００３４】上述のアクセス制御手段を実行するために、本発明は各列にマスク機能を与える。こうして、アクセス手段が、ある条件の下で列へのアクセスを拒否する場合には、その列はマスクされる（隠される）べきである。それ故に、本発明によれば、マスク機能はその列に与えられる。なお、列が現在のアクセス手段で隠されていないが、将来のアクセス手段で隠される場合には、その列にマスク機能を与えることもできる。

【００３５】図１０は、患者名列、即ちＰ＿ＮＭ用のマスク機能１０００の説明例を示している。本発明の実施の形態によれば、マスク機能は、時には「格納された手続き」、「手続きコール」等と呼ばれるユーザ定義機能コール用の従来のＳＱＬタイプ構文によって定義されている。理解されることは、マスク機能の概念は他の方法で実行されることがある。例えば、マスク機能能力を含むようにＳＱＬ言語を再定義することができる。しかしながら、ユーザ定義機能を使用することにより、カスタムＳＱＬ言語に備える必要がなくなる利点がある。

【００３６】マスク機能１０００は一つ以上のキーパラメータ１００２の関連組を含んでいる。また、マスク機能は、関連される本来の値のパラメータ１００４を有している。説明されるように、一つ以上のキーパラメータは、マスクされた列を表示するか否か、又は、列をマスクするか否かを決定する基礎を形成している。図１０で示された例では、Ｐ＿ＮＭ１０００用のマスク機能の中に、二つのキーパラメータ、即ち、ＫＥＹ＿ＰＴ＿ＩＤとＫＥＹ＿ＶＳＴ（１００２）とが存在している。なぜならば、役割ＩＩのユーザ用のアクセス手段が、患者往診によって患者プライベート情報を保護するのに必要とされるからである。そして、ＰＴ＿ＩＤ及びＶＳＴが患者往診対象用のキー列になる。

【００３７】マスク機能はＩＦ＿ＴＨＥＮ＿ＥＬＳＥクローズ１００６を含んでいる。ＩＦ条件は、アクセス手段条件の論理１００８を構成し、それは効果の点でアクセス手段に従って定義される。アクセス手段条件の論理はキーパラメータ１００２の関数である。アクセス手段条件がＴＲＵＥと評価する場合には、マスク機能は列の値として本来の値のパラメータ１００４をリターンする。アクセス手段条件がＦＡＬＳＥと評価する場合には、デフォールト値が列の値としてリターンされる。

【００３８】図１０で示される本発明の実施の形態では、デフォールト値は機能コール１０１０によって作成される。この特別な説明例では、デフォールト値は、本来の値のパラメータ１００４における幾つかの関数である。他の実施の形態では、デフォールト値は、本来の値

のパラメータに限定されない情報に基づくことがある。本発明の更に別の実施の形態では、デフォールト値は、一定の入力、例えば、ＮＵＬＬ、又は、「許可されないアクセス」等のようなテキスト手段にすることがある。動作条件やセキュリティ配慮等は、デフォールト値がどのように決定されるのかを決定する。

【００３９】一般的な形式では、本発明の一実施の形態によるマスク機能は次の構文を有する：

【００４０】

【式１】

$$rv \Leftarrow mask\_name(kp_1, kp_2, \ldots kp_n, op),$$

ここに、ｒｖはマスク機能におけるリターンの列値であり、kp_1，kp_2，・・・，kp_nは、マスクが起こる＊か否かを決定するために利用されるキーパラメータであり、ｏｐは、マスクされた列における本来の値である。

【００４１】機能コール及びその定義における具体的な構文は、一つのＳＱＬ実行から別のＳＱＬ実行に変化している。そのような詳細は、データベース技術における通常の技術を有する者にとって公知であり、且つ、理解される。

【００４２】以下の表Ｉは本発明による一般的なマスク機能の一例である。また、本発明によるフィルタ機能が示されている。

【００４３】

【表１】

表Ｉ

```
/* ================================================== */
/* PACKAGE MASK                                       */
/* ================================================== */

CREATE OR REPLACE PACKAGE FINVIEW.MASK AS
  FUNCTION P_NM(KEY_PT_ID NUMBER, KEY_VST NUMBER, ORG_P_NM
VARCHAR2)
    RETURN VARCHAR2;
  FUNCTION D_NM(KEY_MD_ID NUMBER, ORG_D_NM VARCHAR2)
    RETURN VARCHAR2;
END MASK;


CREATE OR REPLACE PACKAGE BODY MASK IS

  FUNCTION P_NM(KEY_PT_ID NUMBER, KEY_VST, ORG_P_NM VARCHAR2)
    RETURN VARCHAR2
  IS
  BEGIN
    IF FILTER.PT(KEY_PT_ID, KEY_VST)=1 THEN
      RETURN ORG_P_NM;
    ELSE
      RETURN MASKED.P_NM(ORG_P_NM);
    END IF;
  END P_NM;

  FUNCTION D_NM(KEY_MD_ID NUMBER, ORG_D_NM VARCHAR2)
    RETURN VARCHAR2
  IS
  BEGIN
    IF FILTER.MD(KEY_MD_ID)=1 THEN
      RETURN ORG_D_NM;
    ELSE
      RETURN MASKED.D_NM(KEY_MD_ID, ORG_D_NM);
    END IF;
  END D_NM;

END MASK;


/* ================================================== */
/* PACKAGE FILTER                                     */
/* ================================================== */

CREATE OR REPLACE PACKAGE FILTER AS

  FUNCTION PT(KEY_PT_ID NUMBER, KEY_VST NUMBER)
    RETURN NUMBER;
```

```
15
 FUNCTION MD(KEY_MD_ID NUMBER, KEY_VST NUMBER)
   RETURN NUMBER;

END FILTER;

CREATE OR REPLACE PACKAGE BODY FILTER IS

  FUNCTION PT(KEY_PT_ID NUMBER, KEY_VST NUMBER)
    RETURN NUMBER
  IS
    CNT          NUMBER;
  BEGIN
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 1                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_1') = 1 THEN
      RETURN 1;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 2                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_2') = 1 THEN
      EXECUTE IMMEDIATE
      'SELECT COUNT(*) '||
      '  FROM '||SYS_CONTEXT('userenv',
                             'session_user')||'.ACCS_PTVST '||
      ' WHERE PT_ID = :KEY_PT_ID AND VST=:KEY_VST'
       INTO CNT USING KEY_PT_ID, KEY_VST ;
      IF CNT > 0 THEN
           RETURN 1;
      ELSE  RETURN 0;
      END IF;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 3                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_3') = 1 THEN
      RETURN 1;
    END IF;
  END PTVST;

  FUNCTION MD(KEY_MD_ID NUMBER)
    RETURN NUMBER
  IS
    CNT          NUMBER;
  BEGIN
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 1                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_1') = 1 THEN
      RETURN 1;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 2                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_2') = 1 THEN
      EXECUTE IMMEDIATE
      'SELECT COUNT(*) '||
      '  FROM '||SYS_CONTEXT('userenv',
                             'session_user')||'.ACCS_MD '||
      ' WHERE MD_ID = :KEY_MD_ID'  INTO CNT USING KEY_MD_ID;
```

```
17                                                    18
    IF CNT > 0 THEN
          RETURN 1;
    ELSE  RETURN 0;
    END IF;
  END IF;
  /* ------------------------------------------------- */
  /* FOR USER_ROLE_TYP = 3                             */
  /* ------------------------------------------------- */
  IF SYS_CONTEXT('SECURITY', 'ROLE_3') = 1 THEN
    RETURN 1;
  END IF;
 END MD;
END FILTER;
```

示されたマスク機能は、本発明の一般的な実施の形態を例示するためだけに提供されている。付加的なマスク機能は、データベースの複雑性に依存して必要とされることがある。具体的な実行は、使用中のプログラミング言語に依存するだろう。具体的なアルゴリズムは、効果の点で、アクセス手段の特定の要件に依存して変化する。データベース技術における通常の技術を有する者は、特別なデータベースシステムの具体化におけるコンテキストの中で本発明を実行する方法を容易に理解するだろう。

【００４４】また、表１は、ＦＩＬＴＥＲパッケージ内で定義されたフィルタ機能を示している。二つの関数、即ち、ＰＴ０及びＭＤ０が与えられている。ＰＴ０関数は、パラメータＫＥＹ＿ＰＴ＿ＩＤとＫＥＹ＿ＶＳＴとを有している。データをマスクするべき場合には、それが０をリターンする。そして、キーパラメータとユーザ役割とに基づいて、データを表示することがある場合には、それが１をリターンする。この実行では、各々の役割ＩＩのユーザは、自分が処置した患者往診の全てに対して｛ＰＴ＿ＩＤ＿ＶＳＴ｝のリストを維持する表ＰＴＶＳＴを有する。

【００４５】ＭＤ０フィルタ関数は、パラメータＫＥＹ＿ＭＤ＿ＩＤを有する。ＰＴ関数と同様な方法で、それは０又は１をリターンする。ＭＡＳＫ機能はＭＡＳＫパッケージ内で定義されている。この例だけが、Ｐ＿ＮＭ及びＤ＿ＮＭ用のマスク機能を有している。最初に、Ｐ＿ＮＭのマスク機能が、手段関数ＦＩＬＴＥＲ．ＰＴをコールする。それから、結果が１である場合には、それは本来の値ＯＲＧ＿ＰＮＭをリターンする。そして、結果が０である場合には、ＭＡＳＫＥＤ．ＰＮＭ関数によって生成されたマスク値をリターンする。Ｄ＿ＮＭはＰ＿ＮＭと同様な方法で生じる。なお、任意のパラメータを定義してマスク値を作り出すことができる。この例で＊は、ＮＡＳＫＥＤ．Ｐ＿ＮＭはＯＲＧ＿Ｐ＿ＮＭだけを使用し、そして一方、ＭＡＳＫＥＤ．Ｄ＿ＮＭはＫＥＹ＿ＭＤ＿ＩＤとＯＲＧ＿ＤＮＭとの両方を使用している。

【００４６】図１１及び図１２に関して、本発明によるセルレベルのアクセス制御アーキテクチャにおける説明的な実施の形態が示されている。図１１は、レポートテンプレート７３４（図７）の一つの中で一般的に見出される問い合わせ１１０２を示している。その問い合わせは、従来のＳＱＬ構成体を使って書かれている。一般的なＳＱＬ問い合わせは、ＳＥＬＥＣＴ文を含み、一つ以上の列基準（時には属性やフィールド等と呼ばれる）を具体化し、そのＳＥＬＥＣＴ文はその問い合わせの結果を構成している。

【００４７】本発明によれば、翻訳手続き１２１０は、レポートテンプレートからなる問い合わせに適用されて変更レポートテンプレート７３４′を作成する。変更レポートテンプレートからなる問い合わせ１２０２は本来の問い合わせ１１０２の翻訳であって、ある列基準がマスク機能と置き換えられる。

【００４８】翻訳手続き１２１０は、効果の点でアクセス手段に基づいている（例えば、図８）。理解されるように、本来の問い合わせ１１０２は、翻訳された問い合わせ１２０２に非常に類似している。アクセス手段が、マスクされる列基準を要求する場合には、その列基準はマスク機能への適切な機能コールと置き換えられる。

【００４９】例えば、本来の問い合わせ１１０２を検討する。ここで、アクセス手段がマスキングするのに必要である列は、ＰＴ＿ＩＤ、ＶＳＴ、Ｐ＿ＮＭ、ＭＤ＿ＩＤ、及び、Ｄ＿ＮＭ（図９）である。以下の表ＩＩは代用スキームを示している。

【００５０】

【表２】

表ＩＩ

列基準	マスク機能の代用
PT_ID	MASK.PT_ID(c.PT_ID, i.VST) PT_ID
VST	MASK.VST(i.PT_ID, i.VST) VST
P_NM	MASK.P_NM(i.PT_ID, i.VST, p.P_NM) P_NM
MD_ID	MASK.MD_ID(i.MD_ID) MD_ID
D_NM	MASK.D_NM(i.MD_ID, m.D_NM) D_NM

なお、表又はビューＩＤは、各問い合わせのＦＲＯＭクローズに従って、適切な名前に変更されるべきである。例えば、「ｃ．」、「ｉ．」、「ｐ．」、及び「ｍ．」が変更されるべきである。理解されるように、翻訳処理１２１０は、対応する機能コールによって、マスクされた列基準における本来の問い合わせの中で、単にテキスト代用になる。表ＩＩ内に含まれる情報をテキストエディターと共に使用して、図１２で示される翻訳問い合わせ１２０２を作成できる。翻訳処理はスタンダードエディターにすることがある。即ち、テキストエディターを流すユニックスは、特に、利用可能である。翻訳処理は、カスタムな一つのソフトウェアに、又は、ハードウェアとソフトウェアとの何らかの組合せにさえもすることがある。本発明によって、要求される翻訳タスクは、多くの従来技術のうち幾つかを使って提供されることがある。

【００５１】図１２を続けると、翻訳処理１２１０は、本来の問い合わせ１１０２を翻訳問い合わせ１２０２に変換する。それから、翻訳された問い合わせは、ＤＢＭＳ７０２に送られ、その中でその問い合わせが実行される。ＤＢＭＳは一組のユーザ定義機能１２１２を含んでいる。マスク機能定義１２２２はこのユーザ定義機能内に含まれている。

【００５２】また、図１２は、ユーザ定義機能内での一組のフィルタ機能１２２４を示している。フィルタ機能はマスク機能と同じ方法で機能する。マスク機能がアクセス手段に従って列をマスクするのに利用できる場合には、フィルタ機能は、行削除手段毎に、行（記録）をマスクするのに利用できる。フィルタ機能は、行が保持されるべきか又は削除されるべきかを決定する一つ以上のキーパラメータを必要としている。本発明の実施の形態では、フィルタ機能はＴＲＵＥ／ＦＡＬＳＥのようなバイナリー値をリターンする。それをＳＱＬ問い合わせのＷＨＥＲＥクローズ内で使って、行削除手段に従ってリターンされた行を制限する。フィルタ機能１２０４の一例が図１２に示されている。

【００５３】開示された実施の形態は、関係型データベースとＳＱＬタイプの問い合わせ言語とに基づいている。しかしながら、他のデータベースシステム内でマスク及びフィルタ機能処理を提供できることは、データベース技術において通常の技術を有する者によって理解され得る。関係型データベースシステムでは、本発明が、基本的なデータベースエンジンへの影響なく、セルレベルデータのアクセス制御を提供できる。

【００５４】翻訳処理１２１０は、既存のレポートテンプレートを変更する退屈で間違えやすいタスクを除去する。それぞれの問い合わせがデータベースに送られると、翻訳処理が急いで行われることがある。本発明における別の実施の形態では、全てのテンプレート上で翻訳処理を一回処理して（例えば、データベース管理者によって手動で実行される）、マスク及びフィルタ機能を使う新しい一組のテンプレートを作成できる。レポートテンプレートが変更される時だけ翻訳が実行される必要があるので、この実施の形態はスループットの観点から魅力的である。本発明における更に別の実施の形態では、翻訳処理はＤＢＭＳ７０２に配置されているので、全ての入ってくる問い合わせをインターセプトし、急いで翻訳することができる。翻訳処理は、手動で実行されるタスクにすることがある。具体的な処理は、性能基準、問題解決能力、データベース使用の特質、レポート数等に基づいて決定される。

【００５５】マスク機能がＤＢＭＳ内に格納されているので、アクセス手段の変化は、結果的に、マスク及びフィルタ機能における単純なリライトになる。既存のアプリケーション論理に影響を及ぼす必要がない。どの列がマスクされるべきかをアクセス手段が変える場合には、それに従って翻訳処理１２１０が更新される。例えば、マスク列としてＡＧＥ列を加えたい場合には、図１２の本来のＳＱＬが、ＡＧＥ列をマスク機能１３０２と取り替えることによって、図１３で示されるように変更されることがある。

【００５６】本発明の具体的な実施の形態が説明されたけれども、様々な変更、修正、代替構成、及び相当物もまた本発明の範囲内に包含される。説明された発明は、ある特定のデータ処理環境の範囲内で作動するように制限されないが、複数のデータ処理環境の範囲内で自由に作動することができる。本発明が具体的な実施の形態に関して説明されたけれども、本発明の範囲が上述された具体的な実施の形態に限定されないことは当業者にとって明らかである。

【００５７】従って、明細書及び図面は、限定的な意味よりむしろ説明的な意味で考えられるべきである。しか

しながら、追加、削除、代用、及び他の変更は、請求項による本発明の幅広い趣旨及び請求の範囲から逸脱しない範囲内でなすことができる。

【００５８】

【発明の効果】本発明は、それぞれアクセス制御された列のためのマスク機能を使ってセルレベルのアクセス制御を提供する。各々のマスク機能は、アクセス許可を決定する一つ以上のキーパラメータに関連づけられている。マスク機能の中で具体化されるアクセス手段に依存して、マスク機能は、マスクされた列の値、又は、本来の列の値をリターンする。

【００５９】また、それぞれの行削除手段のためのフィルタ機能を使って、セルレベルのアクセス制御を提供する。各々のフィルタ機能は、一つ以上のキーパラメータに関連づけられている。フィルタ機能は２のカテゴリー（例えばバイナリー）値をリターンする。フィルタ機能のリターン値をチェックする条件は、問い合わせの条件クローズに加えられて、行削除手段に従って行を削除する。

【００６０】更に、上述のセルレベルのアクセス制御メカニズムを提供する。

【図面の簡単な説明】

【図１】病院関連データ用のデータ構成における一例を示す図である。

【図２】図１で示され、医者によって一般的に必要とされるデータのビューを示す図である。

【図３】図２で示されたビューを作成するビュー定義を示す図である。

【図４】集約したＳＱＬ文を示す図である。

【図５】従来のビュー定義により定義されたビューの集約問い合わせ結果を示す図である。

【図６】集約した従来技術のビュー定義を示す図である。

【図７】本発明と共に採用可能な、ウェブに基づくレポーティングシステムのアーキテクチャを示す図である。

【図８】データアクセス手段の一般的な一例を示す図である。

【図９】データベースシステム内における表スキームの実例を示す図である。

【図１０】本発明によるマスク機能の実例テンプレートを示す図である。

【図１１】変形前のＳＱＬを示す図である。

【図１２】本発明の一実施の形態におけるセルレベルのアクセス制御のアーキテクチャの外観を示す図である。

【図１３】アクセス手段への変更がどのように本発明内に容易に適合され得るかを示す図である。

【符号の説明】

７００	ウェブによるレポーティングシステムのアーキテクチャ
７０２	データ管理システム
７０４	ウェｂサーバー
７０６	レポートテンプレート
７１２	ブラウザ
７２２	データベースサーバー
７２６	第３コンピュータシステム
７３４	レポートテンプレート
７３４′	変更テンプレート
１１０２	本来の問い合わせ
１２０４	フィルタ機能
１２１２	ユーザ定義機能
１２２２	マスク機能

【図１】

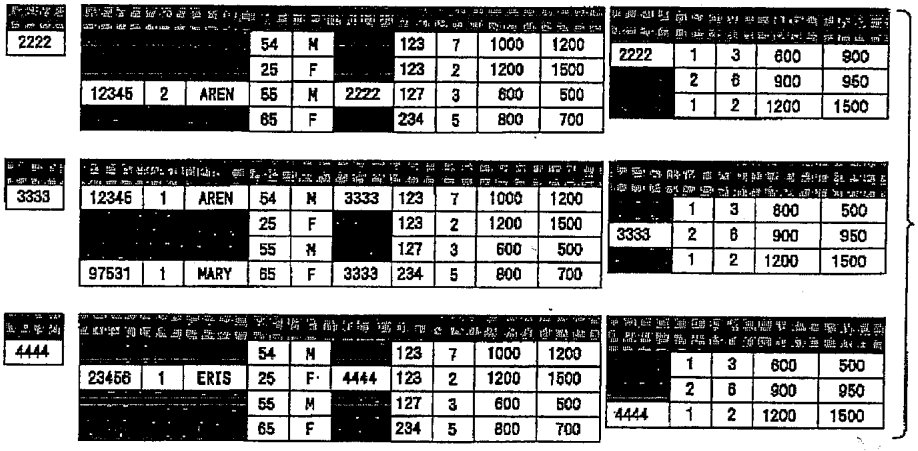
図 1

12345	1	AREN	54	M	3333	123	7	1000	1200
23456	1	ERIS	25	F	4444	123	2	1200	1500
12345	2	AREN	55	M	2222	127	3	600	500
97531	1	MARY	85	F	3333	234	5	800	700

2222	1	3	600	500
3333	2	6	900	950
4444	1	2	1200	1500

【図2】

図 2



2222			54	M		123	7	1000	1200
			25	F		123	2	1200	1500
12345	2	AREN	55	M	2222	127	3	600	500
			65	F		234	5	800	700

2222	1	3	600	900
	2	6	900	950
	1	2	1200	1500

3333									
12345	1	AREN	54	M	3333	123	7	1000	1200
			25	F		123	2	1200	1500
			55	M		127	3	600	500
97531	1	MARY	65	F	3333	234	5	800	700

	1	3	600	500
3333	2	6	900	950
	1	2	1200	1500

4444									
			54	M		123	7	1000	1200
23456	1	ERIS	25	F	4444	123	2	1200	1500
			55	M		127	3	600	500
			65	F		234	5	800	700

	1	3	600	500
	2	6	900	950
4444	1	2	1200	1500

【図3】

図 3

```
CREATE VIEW INPT_FACT AS
SELECT(CASE WHEN MD_ID = user-id THEN PT_ID ELSE NULL END)PT_ID,
   (CASE WHEN MD_ID = user-id THEN VST   ELSE NULL END) VST,
   (CASE WHEN MD_ID = user-id THEN P_NM ELSE NULL END) P_NM,
   AGE,SEX,
   (CASE WHEN MD_ID = user-id THEN MD_ID ELSE NULL END) MD_ID,
   DRG,STAY,COST,FYMT
FROM INPT_BASE;
```

【図11】

図 11

1102

```
SELECT c.PT_ID,
          i.VST,
          p.P_NM,
          p.AGE, p.SEX,
          i. MD_ID,
          m. D_NM,
          i. DRG, i. STAY,…
FROM INPT_FACT i, MD_FACT m, PT_FACT p
WHERE i. PT_ID = p. PT_ID AND i. MD_ID = m. MD_ID AND …;
```

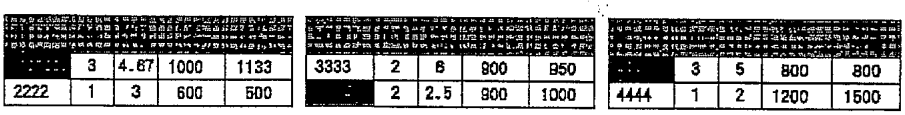
【図4】

図 4

```
SELECT MD_ID, COUNT(*) VOL,
   AVG(STAY) AVG_STAY,AVG(COST) AVG_COST, AVG(FYMT)AVG_FYMT
FROM INPT_FACT
GROUP BY MD_ID ORDER BY AVG_STAY DESC;
```

【図5】

図 5



	3	4.67	1000	1133
2222	1	3	600	500

3333	2	6	900	950
	2	2.5	900	1000

	3	5	800	800
4444	1	2	1200	1500

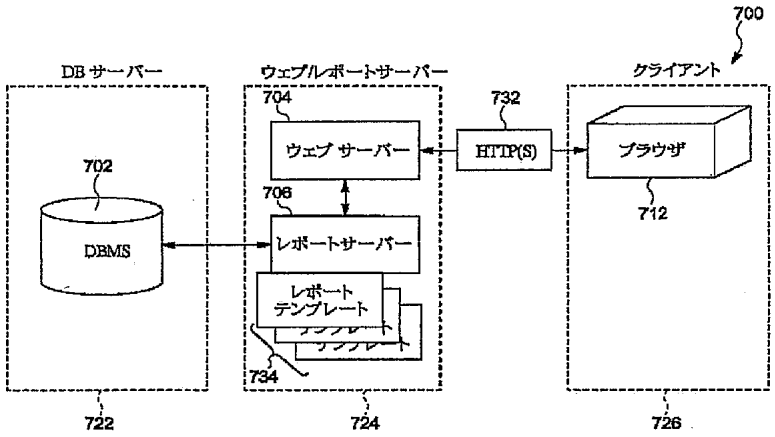
【図6】

図 6

```
CREATE VIEW INPT_GRP_BY_MD
SELECT(CASE WHEN MD_ID = user-id THEN MD_ID ELSE NULL END) MD_ID,
   COUNT(*) VOL,
   AVG(STAY) AVG_STAY, AVG(COST) AVG_COST, AVG(PYMT) AVG_PYMT
 FROM INPT_BASE GROUP BY MD_ID ORDER BY AVG_STAY DESC;
```

【図7】

図 7



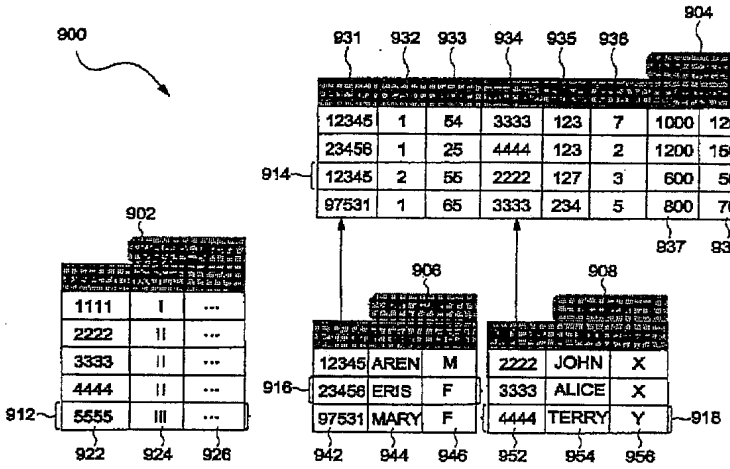
【図8】

図 8

アクセスレベル	役割	アクセス手段
I	実行	全データへのアクセス。
II	医療ドクター	ドクター自身の患者往診データのみへのアクセス。他の患者往診データの患者プライバシー情報は隠されるべきである。他の医療ドクターのプライバシー情報も隠されるべきである。
III	財務アナリスト	医療ドクター情報以外の財務情報へのアクセス。

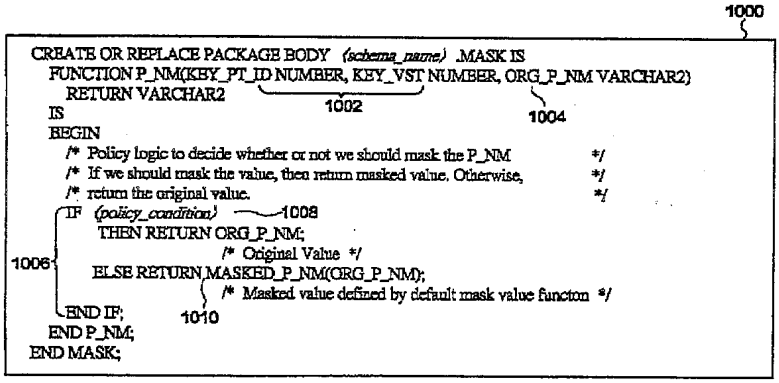
【図9】

図 9



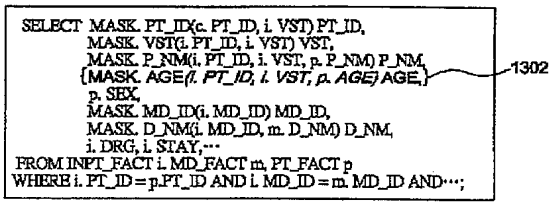
【図10】

図 10



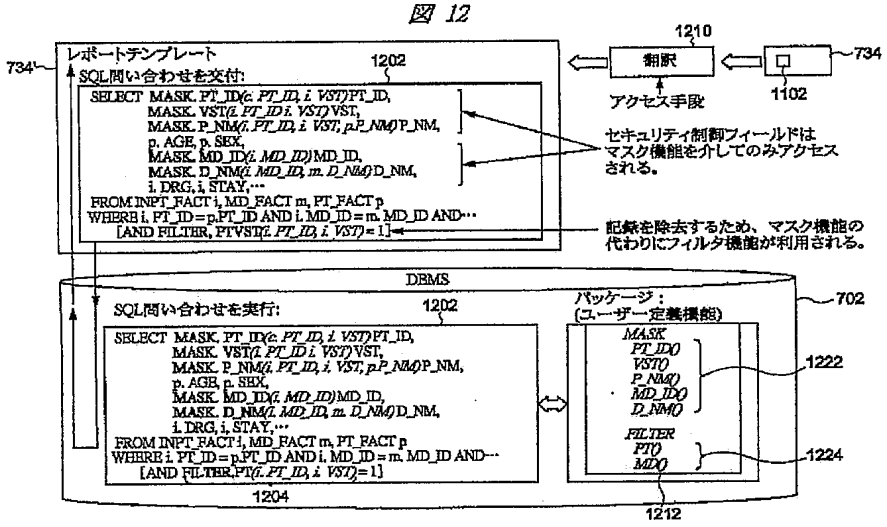
【図13】

図 13



【図１２】

図 12



フロントページの続き

(72)発明者　ミシェル・エル・ケラー
アメリカ合衆国、イリノイ州　60544、プレインフィールド、ビクトリア・ドライブ　4503

Ｆターム(参考)　5B075 KK62 QT06
5B082 GA11

【外国語明細書】

1 Title of Invention

Cell-Level Data Access Control Using User-Defined Functions

2 Claims

1. A method for accessing information in an information store in accordance with an access policy, said method comprising:

receiving an access request comprising a request for a first type of information, wherein said request for a first type of information has associated therewith first information contained in said information store;

replacing said request for a first type of information with a modified request for a first type of information, said modified request being based on said access policy; and

accessing said information store to produce a result in response to said access request, wherein said modified request produces either a masked value or said first information, based on said access policy.

2. The method of claim 1 wherein said modified request includes a mask function.

3. The method of claim 2 wherein said accessing includes executing said mask function to produce either said masked value or said first information.

4. The method of claim 1 further including modifying said access request to include a filter function, said filter function effective for eliminating portions of said result in accordance with said access policy.

5. The method of claim 1 wherein said information store is a relational database and said request for a first type of information comprises a SELECT statement, said SELECT statement comprising one or more column references, said modified request comprising a replacement of at least one of said one or more column references with a mask function.

6. The method of claim 1 wherein said information store is a relational database and said access request includes a WHERE clause, said result comprising one or more rows of information, said method further including incorporating a filter function in said WHERE clause to remove certain rows contained in said result, based on said access policy.

7. In a relational database, a method for accessing information in accordance with an access policy, said method comprising:

providing at least one query comprising a SELECT statement, said SELECT statement comprising one or more column references;

replacing at least one of said one or more column references with a mask function to produce a modified query; and

producing a query result in response to said modified query comprising one or more rows of information;

wherein said query result includes, for said at least one of said one or more column references, either mask values or information from said relational database, based on said access policy.

8. The method of claim 7 wherein said at least one query further comprises a WHERE clause, said method further including modifying said WHERE clause to produce a modified WHERE clause which includes a filter function, said filter function producing one of two logical values, said modified WHERE clause effective for deleting a row from said query result based on a value produced by said filter function.

9. The method of claim 7 wherein said relational database in provided in a database server; said step of providing includes receiving said at least one query at a client system; and said step of producing includes transmitting said modified query to said database server.

10. The method of claim 9 wherein said step of replacing is performed at said client system.

11. The method of claim 9 wherein said step of replacing is performed at said database server.

12. A computer-based information retrieval system comprising:

computer memory having computer readable program code embodied therein for accessing an information store in accordance with an access policy, said computer readable program code comprising:

first code configured to receive an access request for a first type of information, wherein said request for a first type of information has associated therewith first information;

second code configured to replace said request for a first type of information with a modified request for a first type of information, said modified request being based on said access policy; and

third code configured to access said information store to produce a result in response to said access request, wherein said modified request produces either a masked value or said first information, based on said access policy.

13. The system of claim 12 further including fourth code configured to modify said access request to include a filter function, said filter function effective for eliminating portions of said result in accordance with said access policy.

14. The system of claim 12 further including a relational database and said request for a first type of information comprises a SELECT statement, said SELECT statement comprising one or more column references, said modified request comprising a replacement of at least one of said one or more column references with a mask function.

15. The system of claim 12 further including a relational database and said access request includes a WHERE clause, said result comprising one or more rows of information, said second code further configured to incorporate a filter function in said WHERE clause to remove certain rows contained in said result, based on said access policy.

16. The system of claim 12 further including a client computer system and a server computer system, said client computer system comprising a portion of said computer memory embodying said first and second codes, said server computer system comprising another portion of said computer memory embodying said third code.

17. The system of claim 12 wherein said database server is a relational database server, said request for a first type of information comprises a SELECT statement, said SELECT statement comprising one or more column references, said modified request comprising a replacement of at least one of said one or more column references with a mask function.

18. The system of claim 17 wherein said third code includes mask function.

19. The system of claim 16 wherein said database server is a relational database server, said access request includes a WHERE clause, said result comprising one or more rows of information, said second code further configured to incorporate a filter function in said WHERE clause to remove certain rows contained in said result, based on said access policy.

20. The system of claim 19 wherein said third code includes mask function.

3 Detailed Description of Invention

BACKGROUND OF THE INVENTION

The present invention relates generally to database access and in particular to controlled access to fields in a database.

Today's information technology enables one to experience seamless access to various kinds of data sources. Such technology makes accessible to people increasingly greater amounts of information. However, data sources often contain critical information such as medical records, financial records, and other similar personal information which should be protected from unauthorized access, requiring access privilege of those who desire to access such information. Database systems have evolved to provide a set of data access control functions using view definitions and authorization models.

A view is an information object that allows you to view data in a normal table, but in a different way. It is a logical dynamically defined table comprised of portions of the fixed tables which constitute the database. Views provide a method for looking at data in the underlying tables without having to duplicate the data.

The traditional view can control access to data in the database on either a row-level and/or a column level basis. Fig. 1 shows an example of hospital data INPT_BASE 100 that contains inpatient information and aggregated inpatient information grouped by MD_ID. Assume that each physician is permitted only to see his/her patient visit. Fig. 2 shows the desired views of INPT_BASE 100 for each physician. The PT_ID, VST, P_NM and MD_ID fields are selectively made invisible to protect the privacy of each patient so physicians can only see data for their own patients. Thus, for the doctor whose ID is 2222, the view that should be available to that doctor is the view 202. For the doctor whose ID is 3333, the view is view 204.

A view for the inpatient table can be defined by a conventional view definition (or view creation). For example, Fig. 3 shows a view definition that produces the views 202, 204, 206 shown in Fig. 2. (Note that *user-id* can be replaced with an expression that returns the current user-id, e.g., *SYS_CONTEXT('userenv', 'session_user')*, in the case of an Oracle database system.) However, if we execute the SQL statement in Fig. 4 to get the aggregated inpatient information grouped by MD_ID, each physician will get different results such as shown in Fig. 5.

To get the desired aggregation result shown in Fig. 2, we can define a view shown in Fig. 6. However, we must define all possible combinations of aggregation views to allow *ad-hoc* multi-dimensional analysis. This brute force approach greatly increases the view maintenance cost significantly. For example, if a physician wants to see the statistics of specific a DRG (Diagnostic Related Group) e.g., *DRG BETWEEN 120 and 129*, then we must define a view that aggregates the subset of data grouped by MD_ID separately. Since each physician may want to see a different subset of data, it is almost impossible to prepare this view beforehand.

Current systems solve this issue by implementing access-control policies as a part of the application logic. However, there are multiple applications in a typical system. Consequently, an access policy would have to be implemented in each of the different applications, a task which significantly increases the maintenance cost of the access policy. In cases where legacy software is being used, the effort may be completely frustrated.

Database protection can be obtained through a variety of security measures including: flow, inference, and access control. Access controls in information systems

are responsible for ensuring that all direct access to the system object occurs exclusively according to the models and rules fixed by protection policies. Access controls are enhanced to a content-dependent access control model for database systems. In the conventional view definition based on content-dependent access control model, an access rule can be represented by the tuple (s, o, t, p), which specifies that a subject s has access t to those occurrence of object o for which predicate p is true. An enhancement of the model comprises a six tuple (a, s, o, t, p, f), where a is an authorizer subject who granted s the right (o, t, p), while f is a copy of a flag describing the possibility for s to further transfer (o, t, p) to other objects.

Many security models have been proposed in the prior art literature. The Access Matrix model, Take-Grant model, Action-Entity model, and Wood et al. model are discretionary security models. A user query is checked against the authorizations. If it is allowed, the query accesses the object in a specific access mode. Otherwise the access is denied.

In a paper by Lunt, T. F., Denning, D., Schell, R. R., Heckman, M., and W. R. Shockley, entitled "The SeaView Security Model," IEEE Trans. on Software Engineering , Vol. 16, No. 6 (Jun. 1990), pp. 593-607, a security model known as the Sea View model was proposed to protect security of relational database systems by using two layers: Mandatory Access Control (MAC) model and Trusted Computing Base (TCB) model. Sea View controls multilevel data access by generating virtual multi-level relation instances from physical single-level relations.

Other models include Jajodia-Sandhu's model and Smith-Winslett's model which have been proposed as multilevel security models. Security policies for these models generate virtual multi-level relation instances. These models use a commutative filter that is placed between a database system and applications to implement database security.

Processing a conventional view includes the following typical steps:

1) Authentication.
2) Apply view definitions, i.e., rewrite a query according to view definitions.
3) Optimize the query.
4) Execute the query.
5) Return results.

In the conventional view, access control rules are applied to a query before execution. The query cannot access a column that is not a member of the projection

columns. Furthermore, if a user defines a function that blinds the column value as a projection object, the query cannot access the original value either.

Ferraiolo, David F., Barkley, John F., and Kuhn, D. Richard, in a paper entitled "A Role-Based Access Control Model and Reference Implementation Within a Corporate Intranet," Trans. Inf. Syst. Secur. 2, 1 (Feb. 1999), pp. 34 - 64, describe a role-based access control that gives access privileges based on the concept of user-roles.

The Oracle 8i system has a fine-grain access control using a virtual private database, which is discussed in a white paper by Davidson, Mary A., entitled "Creating Virtual Private Databases with Oracle8*i*," Oracle Magazine, (July 1999). This function enables a database designer to add a selection condition string automatically whenever a user accesses the table. The condition string can be generated based on any value, e.g., context values and session values. However, the condition eliminates the rows that do not satisfy it, and so we cannot mask a subset of the columns in a row.

A security model has been proposed for statistical database systems to prevent statistical inference, in a paper by Chin, F. Y., entitled "Security in Statistical Databases for Queries with Small Counts," ACM Trans. Database System, 3, I (Mar. 1978), pp. 92-104. There are three techniques for inference protection, i.e., conceptual, restriction-based, and perturbation-based techniques, see for example "Database Security," by Castano, Silvana, Fugini, Mariagrazia G., Martella, Giancarlo, and Samarati, Pierangela, Addison-Wesley Publishing Company, (1994) and a paper by Adam, Nabil R. and Worthmann, John C., entitled "Security-control Methods for Statistical Databases: A Comparative Study," ACM Comp. Surveys, Vol. 21, No. 4, (Dec. 1989), pp. 515-556. These techniques suppress the statistical values or restrict a combination of group dimensions. However, the techniques do not provide a function that suppresses a dimension value itself. Therefore, they cannot define an access policy for aggregation results such as shown in Fig. 2.

There is a need for flexible cell-level data access control technique based on access policy. An access policy implementation is needed which can reduce system costs.

SUMMARY OF THE INVENTION

The present invention provides cell-level access control using mask functions for each access controlled column. Each mask function is associated with one

or more key parameters which determine the access permission. The mask function returns a masked column value or an original column value, depending on the access policy embodied in the mask function.

Another aspect of the present invention provides cell-level access control using filter functions for each row elimination policy. Each filter function is associated with one or more key parameters. The filter function returns a two-category (e.g. binary) value. A condition for checking return value of the filter function is added to a condition clause in a query to eliminate rows in accordance with the row elimination policy.

Still another aspect of the invention is a reporting system which provides the foregoing cell-level access control mechanisms.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present invention can be readily understood by considering the following detailed description in conjunction with the accompanying drawings:

Fig. 1 illustrates an example of a data organization for hospital-related data;

Fig. 2 illustrates the views of the data shown in Fig. 1, typically required by physicians;

Fig. 3 shows a view definition which produce the views shown in Fig. 2;

Fig. 4 shows a SQL statement with aggregation;

Fig. 5 shows the result of an aggregation inquiry on a view defined by a conventional view definition;

Fig. 6 shows a prior art view definition with aggregation;

Fig. 7 shows web-based reporting system architecture which can be adapted with the present invention;

Fig. 8 illustrates a typical example of a data access policy;

Fig. 9 shows an illustrative example of a table schema in a database system;

Fig. 10 shows an example template of a mask function according to the invention;

Fig. 11 illustrates an SQL prior to modification;

Fig. 12 shows an overview of the cell-level access control architecture in an embodiment of the invention; and

Fig.13 illustrates how changes to the access policy can be readily accommodated in the present invention.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Referring to Fig. 7, an embodiment of the present invention can be described in connection with a web-based reporting system architecture 700. The architecture comprises three server components: A database server 722 includes a database management system (DBMS) 702. The DBMS can be any conventional database system. In one particular illustrative embodiment, the DBMS is a relational database system. A report server 706 is in communication with the database server over conventional communication facilities, the specifics of which depend on the particular embodiment of the invention. The report server includes a plurality of report templates 734 to facilitate its function of providing report generating services. A web server 704, in communication with the report server, provides client-side access to the DBMS. The web server communicates with the report server over conventional communication facilities, the specifics of which depend on particular embodiment of the invention.

Fig. 7 shows a typical software and hardware configuration of the server components. The database server 722 typically occupies its own computer system, including a high capacity storage subsystem. The report server 706 and the web server 704 are shown residing in another computer system 734, separate from the database server. In practice, the web server and the report server may be comprised of multiple instantiations of web server processes and report server processes to achieve a desired throughput. It is noted that many alternative configurations are possible; e.g., a single computer system can be used to host all three servers components in a small scale operation. In a large installation, each server my occupy its own computer system. Each server may in fact comprise multiple server systems in very large systems in order to provide even greater throughput.

User access to the DMBS 702 is made via a browser client 712, executing on yet a third computer system 726. The browser communicates with the web server 704 using the hypertext transport protocol (HTTP) or HTTP over SSL protocol (HTTPS).

A user will interact with the web server 704 via the browser 712 to obtain

a report. First, a report template 734 is selected. Next, a set of parameters for the template is provided. The web server passes a template identifier corresponding to the user-selected report template along with the user-provided parameters to the report server 706. The report server issues one or more queries associated with the selected report template to the database server 722. After some appropriate interactions between the report server and the database server, the results of the query(ies) are returned to the report server. The report server receives the results and formats them into a presentable form which is then delivered to the user through the web server.

Fig. 8 is an example of a simple data access policy shown merely for illustrative purposes. In this example, assume that three access levels are desired: executive level; medical doctor level, and financial analyst level. An executive level user is allowed to access all of the data. Typically, this systemis for administrator personnel and database management personnel.

A physician would be accorded the privileges of a medical doctor-level user. The physician should be able to access patient data relating to treatment of the patient visit, and data that the physician generates. However, the physician is not allowed to access certain of the patient's private information; e.g. credit card information. Furthermore, a physician is not allowed to access the data of another physician. According to the illustrative access policy described in Fig. 8, a physician cannot see the patient name for the patient visits that were treated by the other physician, even if the physician treated the patient's other visit. For example, physician 2222 cannot see the patient name for the first row in Fig. 2, since the other physician 3333 treated AREN's first visit. Therefore, according to the illustrative access policy given in Fig. 8, even though the physician 2222 treated AREN's second visit, that physician cannot see the name for AREN's first visit. It is understood that there are other access policies which allow access to the data in such a case. It is understood that the present invention can provide for such access policies.

Finally, access control is provided for financial personnel. This class of user is given financial analyst level user access. The financial analyst can access financial information such as stay, cost, and payment, including certain of a patient's financial information. However, a financial analyst should not have access the kind of data needed by a physician.

Referring to Fig. 9, a illustrative example of a data schema 900 for the relational DBMS 702 (Fig. 7) is shown. A user information table 902 (USER_INFO) contains a user record (e.g., user record 912) for each user. Each record includes a user-id field 922 and a role field 924, in addition to other user-related information 926. The role field identifies the access level privileges for each user, per the access policy of Fig. 8.

An inpatient information table 904 (INPT_FACT) maintains an inpatient record (e.g. inpatient record 914) for each visit made by a patient. Consequently, a patient is very likely to have multiple entries in this table, one for each visit. A patient-id field 931 identifies the patient. A patient-visit field 932 (VST) indicates each visit/admission occurrence of a patient. Another field is the medical doctor ID field 934, which contains an identifier of the treating physician.

A patient information table 906 (PT_FACT) contains a patient record for each patient. Each record includes a patient-id field 942 (PT_ID), a patient name field 946 (P_NM), and a patient-sex field 946 (SEX). A similar physician information table 908 (MD_FACT) contains information for each physician. This might include, for example, a medical doctor ID field 952 (MD_ID), a name field 954 (D_NM), and a medical doctor department field 956 (DEPT).

Referring now to Figs. 8 and 9, the effect of the access policy as it relates to the data schema 900 will be described. Consider, for example, role II users. Recall that a role II user is a physician. A physician should only be able to view certain information for only those patients treated by that physician. Thus, it can be seen that the patient age field 933, the DRG field 935, the length of stay field 936, the cost field 937, the payment field 938, the patient-sex field 946, and the medical doctor department field 956 can be viewed by the treating physician. However, the patient-id field 931 and 942, the patient-visit field 932, the patient-name field 944, the medical doctor ID field 934 and 952, and the medical doctor name field 954 should not be available to a physician if that patient visit was not treated by that physician or if that information is the physician's own data (e.g., a physician can see his name). Thus, the result of inquiries to the schema 900 should include all data for those patient visits that were treated by the inquiring physician, and partially masked data for those patient visits that were not treated by the inquiring physician.

The access policy for a role II user as shown in Fig. 8 restricts the access

to the patient private information such as PT_ID, VST_NBR, and P_NM by a patient visit (not by a patient). Therefore, the key set to determine whether the patient private information should be masked or not is the column set {PT_ID, VST_NBR}, since these columns are primary keys for the patient visit object. (If the access policy restricts the access by a patient, the key set is {PT_ID}). As for medical doctor information, a role II user can only access his/her own privacy information. Therefore, the MD_ID, and D_NM will be blinded if it is not his/hers. Therefore, the key set to determine whether the physician's private information MD_ID and D_NM should be masked or not is the column set {MD_ID}.

If a role II user issues the query such as: SELECT * FROM PT_FACT; then, all PT_ID, and P_NM columns should be blinded (masked), because a role II user should not get the patient list in the hospital. A role II user can only make his/her own patient list. To make his/her own patient list he should issue the following query: SELECT DISTINCT a.PT_ID, a.P_NM, a.SEX FROM PT_FACT a, INPT_FACT b WHERE a.PT_ID = b.PT_ID and b.MD_ID = *physician's-id*. In this case, we can determine whether the columns PT_ID and P_NM should be masked or not by using the value of {PT_ID, VST_NBR}, since the query joins the PT_FACT and INPT_FACT. In conclusion, we will not allow to be seen the private data if the key columns of the objects to determine the mask are not covered by the tables in the query.

To implement above access control policy, the present invention provides mask functions for each column. Thus, if the access policy denies access to a column under certain conditions, that column should be masked (blinded). In accordance with the invention, a mask function is therefore provided for that column. Note that if a column is not blinded in current access policy but may be blinded in the future access policy, we can also provide a mask function for the column.

Fig. 10 shows an illustrative example of a mask function 1000 for the patient name column, P_NM. In accordance with an embodiment of the invention, mask functions are defined by conventional SQL-type syntax for user-defined function calls, sometimes referred to as "stored procedures", "a procedure call", and so on. It is understood that the idea of a mask function may be implemented in other ways. For example, the SQL language can be redefined to include mask function capability. The use of user-definable functions, however, has the advantage of not having to provide for a custom SQL language.

The mask function 1000 includes an associated set of one or more key parameters 1002. The mask function also has an associated original value parameter 1004. As will be explained, the one or more key parameters form the basis for deciding whether a masked column will be displayed or whether it will be masked. In the example shown in Fig. 10, there are two key parameters: KEY_PT_ID and KEY_VST (1002) in the mask function for P_NM 1000, since the access policy for a role II user requires to protect patient private information by patient visit, and PT_ID and VST is a key column for the patient visit object.

The mask function includes an IF-THEN-ELSE clause 1006. The IF condition constitutes access policy condition logic 1008, which is defined in accordance with the access policy in effect. The access policy condition logic is a function of the key parameters 1002. If the access policy condition evaluates to TRUE, then the mask function returns the original value parameter 1004 as the column value. If the access policy condition evaluates to FALSE, a default value is returned as the column value.

In the embodiment of the invention shown in Fig. 10, the default value is produced by a function call 1010. In this particular illustrative example, the default value is some function of the original value parameter 1004. In another embodiment, the default value may be based on information not limited to the original value parameter. In yet another embodiment of the invention, the default value can be a fixed output; e.g. NULL, or a text string such as "Unauthorized Access", and so on. The operating conditions, security considerations, and the like will determine how the default value would be determined.

In a general form, a mask function according to one embodiment of the invention has the following syntax:

$$rv \Leftarrow \text{mask\_name}(kp_1, kp_2, \ldots kp_n, op),$$

where rv is the return column value of the mask function,

$kp_1, kp_2, \ldots kp_n$ are the key parameters used to determine whether masking occurs, and

op is the original value of the masked column.

The specific syntax of the function call and its definition will vary from one SQL implementation to another. Such details are known and understood by those of ordinary skill in the database art.

Table I below is an example of a typical mask function according to the invention. Also shown is a filter function according to the present invention.

TABLE I

```
/* ================================================== */
/* PACKAGE MASK                                       */
/* ================================================== */

CREATE OR REPLACE PACKAGE FINVIEW.MASK AS
  FUNCTION P_NM(KEY_PT_ID NUMBER, KEY_VST NUMBER, ORG_P_NM
VARCHAR2)
    RETURN VARCHAR2;
  FUNCTION D_NM(KEY_MD_ID NUMBER, ORG_D_NM VARCHAR2)
    RETURN VARCHAR2;
END MASK;

CREATE OR REPLACE PACKAGE BODY MASK IS

  FUNCTION P_NM(KEY_PT_ID NUMBER, KEY_VST, ORG_P_NM VARCHAR2)
    RETURN VARCHAR2
  IS
  BEGIN
    IF FILTER.PT(KEY_PT_ID, KEY_VST)=1 THEN
      RETURN ORG_P_NM;
    ELSE
      RETURN MASKED.P_NM(ORG_P_NM);
    END IF;
  END P_NM;

  FUNCTION D_NM(KEY_MD_ID NUMBER, ORG_D_NM VARCHAR2)
    RETURN VARCHAR2
  IS
  BEGIN
    IF FILTER.MD(KEY_MD_ID)=1 THEN
      RETURN ORG_D_NM;
    ELSE
      RETURN MASKED.D_NM(KEY_MD_ID, ORG_D_NM);
    END IF;
  END D_NM;

END MASK;

/* ================================================== */
/* PACKAGE FILTER                                     */
/* ================================================== */

CREATE OR REPLACE PACKAGE FILTER AS

  FUNCTION PT(KEY_PT_ID NUMBER, KEY_VST NUMBER)
    RETURN NUMBER;
  FUNCTION MD(KEY_MD_ID NUMBER, KEY_VST NUMBER)
    RETURN NUMBER;

END FILTER;
```

```
CREATE OR REPLACE PACKAGE BODY FILTER IS

  FUNCTION PT(KEY_PT_ID NUMBER, KEY_VST NUMBER)
    RETURN NUMBER
  IS
    CNT          NUMBER;
  BEGIN
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 1                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_1') = 1 THEN
      RETURN 1;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 2                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_2') = 1 THEN
      EXECUTE IMMEDIATE
      'SELECT COUNT(*) '||
      '   FROM '||SYS_CONTEXT('userenv',
                              'session_user')||'.ACCS_PTVST '||
      ' WHERE PT_ID = :KEY_PT_ID AND VST=:KEY_VST'
       INTO CNT USING KEY_PT_ID, KEY_VST ;
      IF CNT > 0 THEN
            RETURN 1;
      ELSE  RETURN 0;
      END IF;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 3                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_3') = 1 THEN
      RETURN 1;
    END IF;
  END PTVST;

  FUNCTION MD(KEY_MD_ID NUMBER)
    RETURN NUMBER
  IS
    CNT          NUMBER;
  BEGIN
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 1                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_1') = 1 THEN
      RETURN 1;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 2                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_2') = 1 THEN
      EXECUTE IMMEDIATE
      'SELECT COUNT(*) '||
      '   FROM '||SYS_CONTEXT('userenv',
                              'session_user')||'.ACCS_MD '||
      ' WHERE MD_ID = :KEY_MD_ID'  INTO CNT USING KEY_MD_ID;
```

```
      IF CNT > 0 THEN
            RETURN 1;
      ELSE  RETURN 0;
      END IF;
    END IF;
    /* ------------------------------------------------- */
    /* FOR USER_ROLE_TYP = 3                             */
    /* ------------------------------------------------- */
    IF SYS_CONTEXT('SECURITY', 'ROLE_3') = 1 THEN
      RETURN 1;
    END IF;
  END MD;
END FILTER;
```

The mask function shown is provided merely to illustrate a typical example of an embodiment of the invention. Additional mask functions may be needed depending on the complexity of the database. The specific implementation will depend on the programming language in use. The specific algorithm with vary depending on the specific requirements of the access policy in force. Persons of ordinary skill in the database arts will readily understand how to practice the invention in the context of a particular database system installation.

Table I also shows a filter function which is defined in the FILTER package. Two functions are provided, PT() and MD(). The PT() function has a parameter KEY_PT_ID and KEY_VST. It returns 0 if the data should be masked and returns 1 if the data can be displayed, based on the key parameters and a user role. In this implementation, each role II user has a table PTVST that keeps the list of {PT_ID, VST} for all patient visits that he/she treated.

The MD() filter function has a parameter KEY_MD_ID. It returns 0 or 1 in the same way as PT function. MASK functions are defined in the MASK package. This example only includes the mask function for P_NM and D_NM. P_NM mask function first calls the policy function FILTER.PT. Then, if the result is 1, it returns the original value, ORG_P_NM, and if the result is 0, it returns the masked value that is generated by MASKED.P_NM function. D_NM does in the same way as P_NM. Note that we can define any parameters to create masked values. In this example, MASKED.P_NM uses only ORG_P_NM, while MASKED.D_NM uses both KEY_MD_ID and ORG_D_NM.

Referring now to Figs. 11 and 12, an illustrative embodiment of a cell-level access control architecture in accordance with the present invention is shown. Fig. 11 shows a query 1102 that would typically be found in one of the report templates 734

(Fig. 7). The query is written using conventional SQL constructs. A typical SQL query includes a SELECT statement, specifying one or more column references (sometimes referred to as attributes, fields, etc.), which constitute the result of the query.

In accordance with the invention, a translation procedure 1210 is applied to queries comprising the report templates to produce modified report templates 734'. The queries 1202 comprising the modified report templates are translations of the original queries 1102, wherein certain column references are replaced with mask functions.

The translation procedure 1210 is based on the access policies in effect (e.g., Fig. 8). As can be seen, the original query 1102 is very similar to the translated query 1202. Where the access policy calls for a column reference to be masked, the column reference is replaced with an appropriate function call to a mask function.

Consider the original query 1102, for example. Here, the columns which the access policy requires masking are: PT_ID, VST, P_NM, MD_ID, and D_NM (Fig. 9). Table II below shows the replacement scheme:

TABLE II

Column Reference	Mask Function Replacement
PT_ID	MASK.PT_ID(c.PT_ID, i.VST) PT_ID
VST	MASK.VST(i.PT_ID, i.VST) VST
P_NM	MASK.P_NM(i.PT_ID, i.VST, p.P_NM) P_NM
MD_ID	MASK.MD_ID(i.MD_ID) MD_ID
D_NM	MASK.D_NM(i.MD_ID, m.D_NM) D_NM

Note that the table or view ID should be modified to the appropriate name, according to the FROM clause of each query. For example "c.", "i.", "p.", "m.", should be modified. As can be seen the translation process 1210 is simply a textual replacement in the original query of the masked column references by their corresponding function calls. The information contained in Table II can be used in conjunction with a text editor to produce the translated query 1202 shown in Fig. 12. The translation process can be a standard editor, e.g., the Unix streaming text editor is especially applicable. The translation process can be a custom piece of software, or even some combination of hardware and software. The translation task called for by the present invention can be provided using any of a number of conventional techniques.

Continuing with Fig. 12, the translation process 1210 converts an original query 1102 into a translated query 1202. The translated query is then transmitted to the DBMS 702, where the query is executed. The DBMS includes a set of user-defined functions 1212. Included in those user-defined functions are the mask function definitions 1222.

Fig. 12 also shows in the user-defined functions a set of filter functions 1224. The filter functions perform in the same manner as the mask functions. Where the mask functions serve to mask out columns in accordance with the access policy, the filter functions serve to mask out rows (records) per a row elimination policy. Filter functions require one or more key parameters that determine whether a row is to be retained or eliminated. In an embodiment of the invention, the filter function returns a binary value such as TRUE/FALSE. It is used in a WHERE clause of an SQL query to limit the rows that are returned in accordance with the row elimination policy. An example of a filter function 1204 is shown in Fig. 12.

The disclosed embodiments are based on relational databases and SQL-type query languages. However, it can be appreciated by a person of ordinary skill in the database art that the mask and filter function approach can be provided in other database systems. In a relational database system, the present invention can provide cell-level data access control with no impact to the underlying database engine.

The translation process 1210 obviates the tedious and error-prone task of modifying existing report templates. The translation process can occur on-the-fly as each query is sent to the database. In another embodiment of the invention, the translation process can be run once (e.g., manually performed by the database administrator) on all of the templates to produce a new set of templates that use the mask and filter functions. This embodiment is attractive from a throughput point of view, since the translation needs to be performed only when a report template is changed. In yet another embodiment of the invention, the translation process can be located at the DBMS 702, intercepting all incoming queries and making the translations on-the-fly. The translation process could be a manually performed task. The specific approach will be determined based on performance criteria, resources, the nature of the use of the database, the number of reports and so on.

Since the mask functions are stored in the DBMS, a change in the access policy amounts to simple re-writing of the mask and filter functions. There is no need to affect the existing application logic. If the access policy changes which columns are to be

masked, then the translation process 1210 would be updated accordingly. For example, if we want to add AGE column as a mask column, the original SQL in Fig. 2 might be changed as shown in Fig. 13 by the replacement of the AGE column with a mask function 1302.

Although specific embodiments of the invention have been described, various modifications, alterations, alternative constructions, and equivalents are also encompassed within the scope of the invention. The described invention is not restricted to operation within certain specific data processing environments, but is free to operate within a plurality of data processing environments. Although the present invention has been described in terms of specific embodiments, it should be apparent to those skilled in the art that the scope of the present invention is not limited to the described specific embodiments.

The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that additions, subtractions, substitutions, and other modifications may be made without departing from the broader spirit and scope of the invention as set forth in the claims.

4 Brief Description of Drawings

Fig. 1 illustrates an example of a data organization for hospital-related data;

Fig. 2 illustrates the views of the data shown in Fig. 1, typically required by physicians;

Fig. 3 shows a view definition which produce the views shown in Fig. 2;

Fig. 4 shows a SQL statement with aggregation;

Fig. 5 shows the result of an aggregation inquiry on a view defined by a conventional view definition;

Fig. 6 shows a prior art view definition with aggregation;

Fig. 7 shows web-based reporting system architecture which can be adapted with the present invention;

Fig. 8 illustrates a typical example of a data access policy;

Fig. 9 shows an illustrative example of a table schema in a database system;

Fig. 10 shows an example template of a mask function according to the invention;

Fig. 11 illustrates an SQL prior to modification;

Fig. 12 shows an overview of the cell-level access control architecture in an embodiment of the invention; and

Fig.13 illustrates how changes to the access policy can be readily accommodated in the present invention.

RFID	NO	NAME	AGE	SEX	MD_ID	DRG	STAY	COST	P/A
12345	1	AREN	54	M	3333	123	7	1000	1200
23456	1	ERIS	25	F	4444	123	2	1200	1500
12345	2	AREN	55	M	2222	127	3	600	500
97531	1	MARY	65	F	3333	234	5	800	700

MD_ID	NO	AVG STAY	AVG COST	AVG P/A
2222	1	3	600	500
3333	2	6	900	950
4444	1	2	1200	1500

FIG. 1

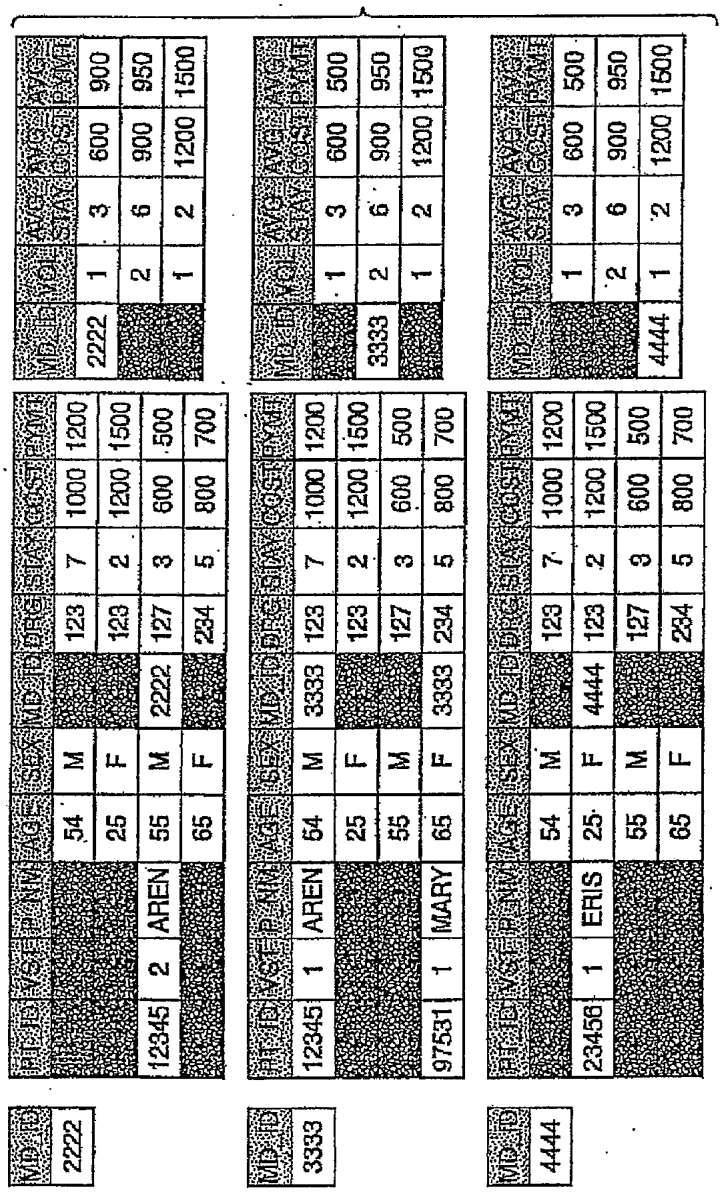

MD_ID
2222
PT_ID VST P_NM AGE SEX MD_ID DRG STAY COST PMT
54 M 123 7 1000 1200
25 F 123 2 1200 1500
12345 2 AREN 55 M 2222 127 3 600 500
65 F 234 5 800 700
MD_ID VST P_NM AGE SEX MD_ID DRG STAY COST PMT
12345 1 AREN 64 M 3333 123 7 1000 1200
25 F 123 2 1200 1500
55 M 127 3 600 500
97531 1 MARY 65 F 3333 234 5 800 700
MD_ID
4444
PT_ID VST P_NM AGE SEX MD_ID DRG STAY COST PMT
54 M 123 7 1000 1200
23456 1 ERIS 25 F 4444 123 2 1200 1500
55 M 127 3 600 500
65 F 234 5 800 700
MD_ID VOL AVG STAY AVG COST AVG PMT
2222 1 3 600 900
2 6 900 950
1 2 1200 1500
MD_ID VOL AVG STAY AVG COST AVG PMT
3333 1 3 600 500
2 6 900 950
1 2 1200 1500
MD_ID VOL AVG STAY AVG COST AVG PMT
4444 1 3 600 500
2 6 900 950
1 2 1200 1500

FIG. 2

```
CREATE VIEW INPT_FACT AS
SELECT (CASE WHEN MD_ID = user-id THEN PT_ID  ELSE NULL END) PT_ID,
   (CASE WHEN MD_ID = user-id THEN VST    ELSE NULL END) VST,
   (CASE WHEN MD_ID = user-id THEN P_NM   ELSE NULL END) P_NM,
   AGE, SEX,
   (CASE WHEN MD_ID = user-id THEN MD_ID  ELSE NULL END) MD_ID,
   DRG, STAY, COST, PYMT
FROM INPT_BASE;
```

FIG. 3

```
SELECT MD_ID, COUNT(*) VOL,
   AVG(STAY) AVG_STAY, AVG(COST) AVG_COST, AVG(PYMT) AVG_PYMT
FROM INPT_FACT
GROUP BY MD_ID ORDER BY AVG_STAY DESC;
```

FIG. 4

MD_ID	I/O	AVG. STAY	AVG. COST/DAY	AVG. COST/MT
2222	1	3	600	500
	3	4.67	1000	1133

MD_ID	I/O	AVG. STAY	AVG. COST/DAY	AVG. COST/MT
3333	2	6	900	950
	2	2.5	900	1000

MD_ID	I/O	AVG. STAY	AVG. COST/DAY	AVG. COST/MT
4444	1	2	1200	1500
	3	5	800	800

FIG. 5

```
CREATE VIEW INPT_GRP_BY_MD
SELECT (CASE WHEN MD_ID = user-id THEN MD_ID ELSE NULL END) MD_ID,
    COUNT(*) VOL,
    AVG(STAY) AVG_STAY, AVG(COST) AVG_COST, AVG(PYMT) AVG_PYMT
  FROM INPT_BASE GROUP BY MD_ID ORDER BY AVG_STAY DESC;
```

FIG. 6

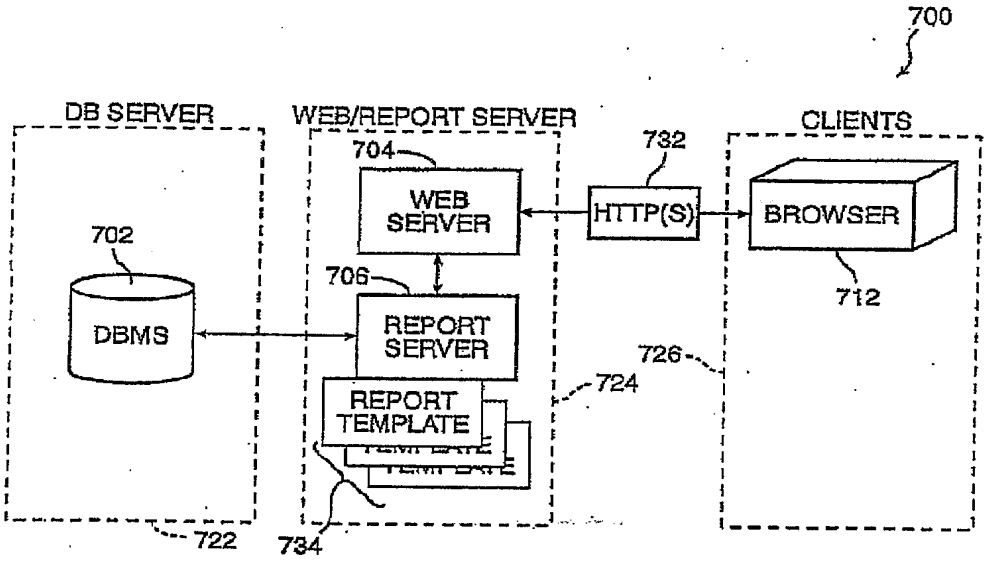


FIG. 7

Access Level	Role	Access Policy
I	Executive	Access to all data.
II	Medical Doctor	Access to doctor's own patient visit data only. The patient privacy information of the other patient visit data should be blinded. The other medical doctor's privacy information should be also blinded.
III	Financial Analyst	Access to financial information without any medical doctor's information.

FIG. 8

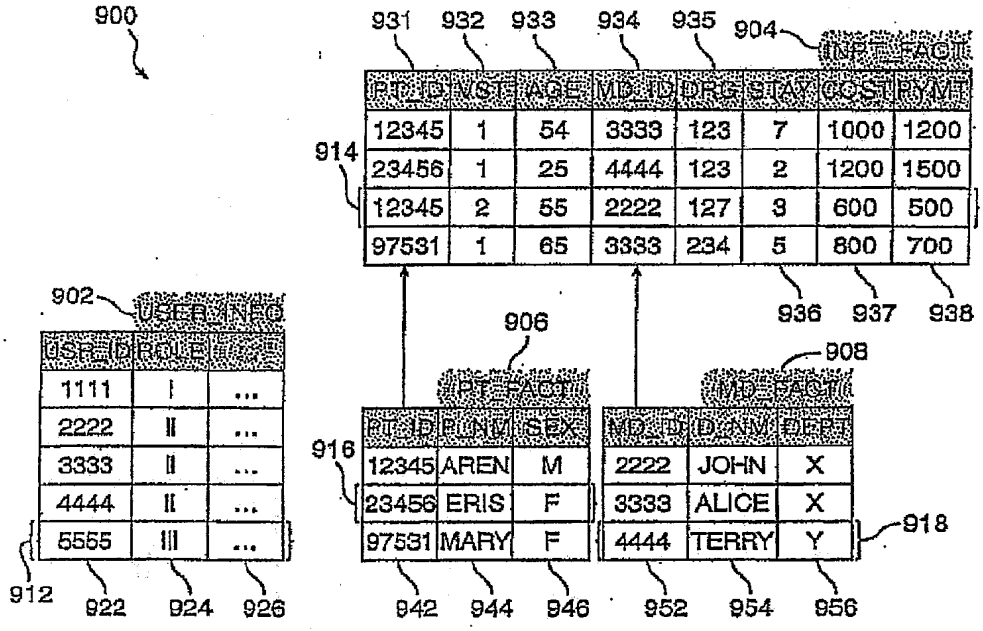


FIG. 9

1000

```
CREATE OR REPLACE PACKAGE BODY <schema_name>.MASK IS
  FUNCTION P_NM(KEY_PT_ID NUMBER, KEY_VST NUMBER, ORG_P_NM VARCHAR2)
    RETURN VARCHAR2
  IS                                   1002          1004
  BEGIN
    /* Policy logic to decide whether or not we should mask the P_NM      */
    /* If we should mask the value, then return masked value. Otherwise,  */
1006 /* return the original value.                                         */
    IF <policy_codition> ~1008
      THEN RETURN ORG_P_NM;
                /* Original Value */
      ELSE RETURN MASKED_P_NM(ORG_P_NM);
                /* Masked value defined by default mask value function */
    END IF;   1010
  END P_NM;
END MASK;
```

FIG. 10

```
SELECT   c.PT_ID,
         I.VST,
         p.P_NM,
         p.AGE, p.SEX,
         I.MD_ID,
         m.D_NM,
         i.DRG, i.STAY, ...
  FROM INPT_FACT I, MD_FACT m, PT_FACT p
WHERE  I.PT_ID=p.PT_ID AND I.MD_ID = m.MD_ID AND ... ;
```

1102

FIG. 11

```
SELECT   MASK.PT_ID(c.PT_ID, i.VST)  PT_ID,
         MASK.VST(i.PT_ID, i.VST) VST,
         MASK.P_NM(i.PT_ID, i.VST, p.P_NM) P_NM,
         MASK.AGE(I.PT_ID, I.VST, p.AGE) AGE,
         p.SEX,
         MASK.MD_ID(i.MD_ID) MD_ID,
         MASK.D_NM(i.MD_ID, m.D_NM) D_NM,
         i.DRG, i.STAY, ...
  FROM INPT_FACT i, MD_FACT m, PT_FACT p
WHERE  I.PT_ID=p.PT_ID AND I.MD_ID = m.MD_ID AND ... ;
```

1302

FIG. 13

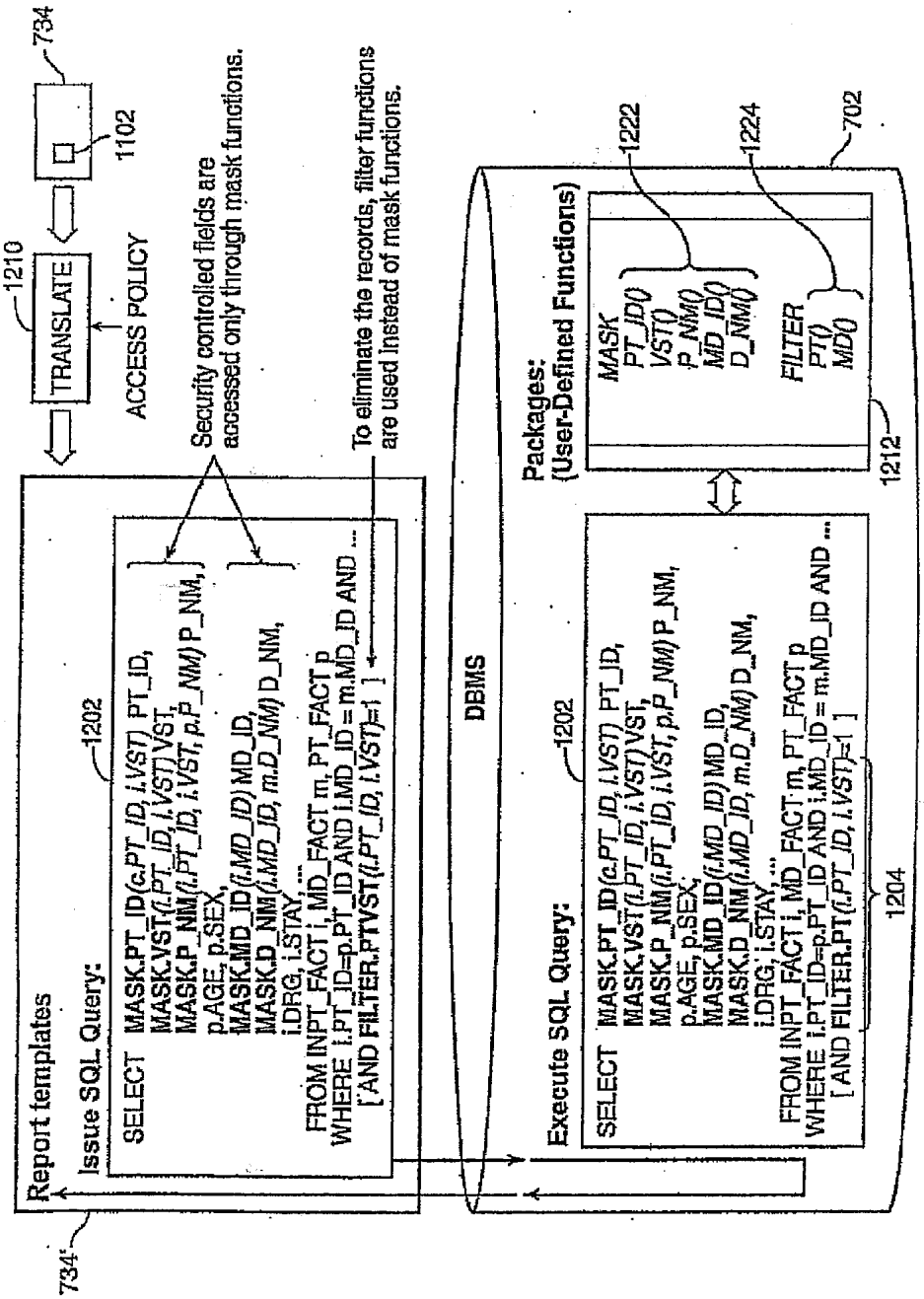

734
1102
1210
TRANSLATE
ACCESS POLICY
Report templates
Issue SQL Query:
1202
SELECT MASK.PT_ID(c.PT_ID, i.VST) PT_ID,
MASK.VST(i.PT_ID, i.VST) VST,
MASK.P_NM(i.PT_ID, i.VST, p.P_NM) P_NM,
p.AGE, p.SEX,
MASK.MD_ID(i.MD_ID) MD_ID,
MASK.D_NM(i.MD_ID, m.D_NM) D_NM,
i.DRG, i.STAY, ...
FROM INPT_FACT i, MD_FACT m, PT_FACT p
WHERE i.PT_ID=p.PT_ID AND i.MD_ID = m.MD_ID AND ...
[AND FILTER.PTVST(i.PT_ID, i.VST)=1]
Security controlled fields are accessed only through mask functions.
To eliminate the records, filter functions are used instead of mask functions.
734'
DBMS
Execute SQL Query:
1202
SELECT MASK.PT_ID(c.PT_ID, i.VST) PT_ID,
MASK.VST(i.PT_ID, i.VST) VST,
MASK.P_NM(i.PT_ID, i.VST, p.P_NM) P_NM,
p.AGE, p.SEX,
MASK.MD_ID(i.MD_ID) MD_ID,
MASK.D_NM(i.MD_ID, m.D_NM) D_NM,
i.DRG, i.STAY, ...
FROM INPT_FACT i, MD_FACT m, PT_FACT p
WHERE i.PT_ID=p.PT_ID AND i.MD_ID = m.MD_ID AND ...
[AND FILTER.PT(i.PT_ID, i.VST)=1]
1204
Packages:
(User-Defined Functions)
1222
MASK
PT_ID()
VST()
P_NM()
MD_ID()
D_NM()
1224
FILTER
PT()
MD()
1212
702

FIG. 12

1 Abstract

Access control at the cell level is provided by the use of mask functions. Original queries are modified to contain mask functions for those cells which controlled access in accordance with an access policy is desired. In addition, filter functions are included to eliminate rows according to the access policy

2 Representative Drawing

FIG. 12